edv aspekte

187
Herausgegeben
von der Redaktion
rechentechnik
datenverarbeitung
DDR 5,00 M

Arbeitsplatz-computer A 7100

Gerätetechnik
Betriebssysteme
Programmier-sprachen
Basis-software

Inhalt

Siegfried Kerst, Harald Riegel:
Arbeitsplatzcomputer robotron A 7100 2

Ulrich Heckel:
Das Betriebssystem SCP 1700 18

Bernd Kubsch:
Das Timesharing – Betriebssystem MUTOS 1700 25

Dr. Peter Kühlborn, Manfred Synowzik:
Das echtzeitorientierte Betriebssystem BOS 1810 33

Dr. Rolf Dottermusch:
Sprachkonzeption des A 7100 39

Ursula Hempel, Hans Loley:
REDABAS – relationales Datenbankbetriebssystem für 8- und 16-Bit-Mikrorechner 45

Rainer Weber:
TEXT 40/M16 – Programmsystem zur Textverarbeitung auf 16-Bit-Bürocomputer 51

Wilfried Heymel:
Softwarebausteine für den AC 7100 56

Dieter Wildenhain:
Robotron-Software für mathematische Verfahren 63

содержание

Зигфрид Керст, Харальд Ригель:
Компьютер для рабочего места роботрон А 7100 2

Ульрих Хекель:
Операционная система СЗП 1700 18

Бернд Кубш:
ОС с разделением времени МУТОС 1700 25

Д-р Петер Кюльборн, Манфред Зиновиц:
итированная на реальное время ОЗ БОС 1810 33

Д-р Рольф Доттермуш:
Языковая концепция ЭВМ А 7100 39

Урзула Хемпель, Ханс Лолей:
ПЕДБАС – релационная ОС банка данных для микро ЭВМ 8- и 16-бит 45

Райнер Вебер:
TEXT 40/M16 – система программ для обработки текста на 16-разрядных персональных компьютерах 51

Вильфрид Хаймель:
Компоненты матобеспечения для АЗ 7100 56

Дитер Вильденхайн:
Математическое обеспечение Роботрон для математических методов 63

In this Issue

Siegfried Kerst, Harald Riegel:
robotron A 7100 workstation computer 2

Ulrich Heckel:
The SCP 1700 operating system 18

Bernd Kubsch:
The MUTOS 1700 timesharing operating system 25

Dr. Peter Kühlborn, Manfred Synowzik:
The BOS 1810 real-time-oriented operating system 33

Dr. Rolf Dottermusch:
The A 7100 language concept 39

Ursula Hempel, Hans Loley:
REDABAS–relational data bank operating system for 8- and 16-bit microcomputers 45

Rainer Weber:
The TEXT 40/M16 program system for text processing on 16-bit office computers 51

Wilfried Heymel:
Soft modules for the AC 7100 56

Dieter Wildenhain:
Robotron software for mathematical processes 63

ISSN 0232-6833 · edv-aspekte · Berlin 6 (1987) 1 · S. 1–64

edv aspekte

6. Jahrgang 1/1987

Verlag Die Wirtschaft Berlin
Am Friedrichshain 22 Berlin 1055
Verlagsdirektor: Dieter Grüneberg

edv-aspekte
Zeitschrift für spezielle Themen
der Informationsverarbeitung,
herausgegeben von der Redaktion
rechentechnik/datenverarbeitung,
Am Friedrichshain 22, Berlin, 1055
Chefredakteur: Franz Loll 4 38 73 41
Sekretariat: 4 38 72 33
Fernschreiber: 114 566
Titelgestaltung: Marlies Hawemann
Redaktionsschluß: 9. September 1986

Lizenz des Presseamtes beim Vorsitzenden des Ministerrates der DDR Nr. 1529

edv-aspekte
Erscheinungsweise vierteljährlich zum Bezugspreis DDR 5,00 M je Heft
EDV-Artikel-Nr. 1331
Auslandspreise sind dem Zeitschriftenkatalog des Außenhandelsbetriebes Buchexport zu entnehmen.

Satz: Verlag Die Wirtschaft, Berlin
Druck: (140) „Neues Deutschland", Berlin

Anzeigenverwaltung:
Berliner Verlag,
Karl-Liebknecht-Str. 25
Berlin 1056 Telefon: 2 70 33 02

Anzeigenannahme:
Berliner Verlag und Annahmestellen in Berlin und in den Bezirken
Zur Zeit gültige Anzeigenpreisliste Nr. 12

Im Ausland:
INTERWERBUNG GmbH – Gesellschaft für Werbung und Auslandsmessen der DDR, Hermann-Duncker-Str. 89 Berlin 1157

Bestellungen nehmen entgegen:
Für die DDR
Sämtliche Postämter, der örtliche Buchhandel und der Verlag Die Wirtschaft Berlin
Inkasso-Zeitraum: vierteljährlich

Im Ausland:
In den sozialistischen Ländern nur der zuständige Postzeitungsvertrieb In allen anderen Staaten der örtliche Buch- und Zeitschriftenhandel. Bestellungen des Buch- und Zeitschriftenhandels sind zu richten an

BUCHEXPORT
Volkseigener Außenhandelsbetrieb der DDR,
DDR – Leninstr. 16 Leipzig 7010, Postfach 160
oder an Verlag Die Wirtschaft,
DDR – Am Friedrichshain 22 Berlin 1055

Mitglieder des Redaktionsbeirates
Dr. Claus Goedecke · Dr. Rolf Gräßler
Prof. Dr. sc. Gerhard Keßler · Dr. Rolf Kilian
Hans Kunau·Walter Münch · Axel Rathsack
Dr. sc. Gerd Rossa · Dr. Claus Sattler · Prof. Dr. sc. Wolfgang Schoppan (Vorsitzender)
Dr. Werner Schulze · Horst Stoll
Prf. Dr. Franz Stuchlik · Dr. Dieter Urban

Gerätetechnik, Betriebssysteme, Programmiersprachen und Basissoftware des Arbeitsplatzcomputers A 7100

Mit dem Arbeitsplatzcomputer A 7100 wird die Entwicklung leistungsfähiger Mikrorechner aus dem VEB Kombinat Robotron fortgesetzt. Er erweitert das Sortiment der bisher als Einzelplatzrechner oder als Terminals produzierten Büro- und Personalcomputer auf 8-bit-Mikrorechnerbasis. Auf Grund des 16-bit-Mikrorechnerschaltkreises ist er für anspruchsvolle Aufgaben, darunter auch für den CAD/CAM-Einsatz, vorgesehen.
Aufgebaut ist der A 7100 auf der Basis des sowjetischen Mikrorechner-Schaltkreissystems K 1810. Seine Logikbaugruppen (Module) sind Funktionseinheiten des neuen Mikrorechner-Modulsystems 16 (MMS 16). Mit der Chiffre CM 1910 gehört er zum System der Kleinrechner (SKR) der sozialistischen Länder. Seine Einsatzgebiete sind u.a.:

- Büroautomatisierung: Planung und Leitung, Rechnungsführung und Statistik, ökonomische Berechnungen, Fakturierung, Optimierungsaufgaben, Terminkontrolle, Datenerfassung, Textverarbeitung;
- Datenbank, Informations- und Auskunftssysteme;
- Ingenieur- und wissenschaftlich-technische Berechnungen;
- Arbeitsplätze für Konstrukteure und Technologen;
- Labor-, Experimental- und Prüffeldautomatisierung;
- Auswerterechner in der Meßtechnik;
- Programmentwicklungen, u. a. für Echtzeitsteuerung.

Der A 7100 kann als Einzelplatzsystem, Rechner im lokalen Netz und Terminal für ESER- und SKR-Rechner in DFV- und Kommunikationssystemen eingesetzt werden. Entsprechend den Einsatzgebieten werden unterschiedliche Betriebssysteme bereitgestellt:

- Das Hauptbetriebssystem SCP 1700 ist ein diskettenorientiertes Einzelnutzersystem. SCP 1700 ist nicht für Echtzeitanwendungen ausgelegt.
- Das Betriebssystem MUTOS 1700 ist als interaktives Teilnehmersystem besonders geeignet für die Programmentwicklung und Sicherung einer einheitlichen Anwenderschnittstelle bei der Realisierung von Anwenderlösungen über verschiedene Rechnerklassen.
- Das Betriebssystem BOS 1810 ist ein Echtzeitbetriebssystem für den A 7100, das auch für den OEM-Einsatz und als Software-Entwicklungssystem genutzt werden kann.

Darüber hinaus erlaubt die realisierte Systemstruktur des A 7100 auch andere Betriebssysteme zu implementieren, wie z. B. MS/DOS.
Um eine effektive Entwicklung multivalent nutzbarer Software zu gewährleisten, werden standardisierte bzw. international eingeführte Programmiersprachen bereitgestellt. Die Sprachkonzeption des A 7100 wird gleichfalls vorgestellt.
Für die Einsatzgebiete Büroautomatisierung, Datenbank-, Informations- und Auskunftssysteme und Ingenieur- und wissenschaftlich-technische Berechnungen wird eine umfangreiche Standardsoftware bereitgestellt. Sie sichert eine breite Anwendung des A 7100 für Nutzer in den Fachabteilungen, die über keine tiefgehenden Kenntnisse in der Programmierung verfügen müssen.
Mit dem Textverarbeitungssystem TEXT 40/M16 sind viele Arbeitsgänge im Büro zu rationalisieren, wie Texte erfassen, korrigieren, aktualisieren und mischen, Schreiben von Serienbriefen, Terminüberwachung und -kontrolle.
Bei der Nutzung dieser Standardsoftware wird der Anwender mit einer komfortablen Menütechnik an die Lösung seiner Probleme herangeführt, ohne daß er Kenntnisse über Programmiersprachen haben muß. Sie stellt damit eine neue Qualität in der Nutzung der Rechentechnik dar und erfordert ein Umdenken gegenüber der traditionellen Einsatzvorbereitung. Vorliegende edv-aspekte-Ausgabe soll bei der Lösung von Problemen beim Einsatz des A 7100 Unterstützung geben.

Dr. Rolf Gräßler

Arbeitsplatzcomputer robotron A 7100

Siegfried Kerst, Harald Riegel
VEB Robotron-Elektronik Dresden

1. Übersicht

Der Arbeitsplatzcomputer A 7100 (AC) ist ein leistungsstarker, grafikfähiger 16-Bit-Mikrorechner auf der Basis des sowjetischen Mikroprozessor-Schaltkreissystems K 1810 mit dem 16-Bit-Mikroprozessor K 1810 WM 86.

Der Einsatz aller systembestimmenden und peripheren Schaltkreise aus dem genannten Mikroprozessor-Schaltkreissystem erlaubt eine kompakte Realisierung seiner Logikbaugruppen (Module), die unter dem Gesichtspunkt entwickelt wurden, daß sie gleichzeitig als abgeschlossene Funktionseinheiten Bestandteile des neuen Mikrorechner-Modulsystems 16 (MMS 16) sind/1/. Als Rechner der Familie M 16-1 gehört der AC A 7100 mit der Chiffre CM 1910 auch zum Bestand des Systems der Kleinrechner (SKR) im Rahmen des RGW.

Der AC A 7100 verwendet den im SKR standardisierten Mikrorechnerbaus 141 als Systembus und realisiert die hauptsächlichsten Standardinterfaces, wobei das international breit eingeführte Subminiatur-D-Steckverbindersortiment zum Einsatz kommt.

In seiner Grundausstattung ist der AC A 7100 ein flexibler, erweiterungs- und anpassungsfähiger Personalcomputer mit einer durchgängigen modularen Gestaltung sowohl in konstruktiver als auch logisch-funktioneller Hinsicht. Für seine Hauptbaugruppen kommen sämtliche Neu- und Weiterentwicklungen zum Einsatz, wobei eine neue Basiskonstruktion bei den Logikmoduln sowie eine neue platzsparende, kompakte Gefäßlösung verwendet werden. Die Realisierung eines universellen standardisierten Bussystems und die durchgängig modulare Gestaltung sind Merkmale des neuen Arbeitsplatzcomputers A 7100, die ihn in die Gruppe der offenen Systeme einordnen.

Individuelle Erweiterungsmöglichkeiten durch Nutzung der zum System gehörenden Baugruppen, mit dem Einsatz neuentwickelter leistungsfähigerer Baugruppen sowie mit Eigenentwicklungen spezieller Baugruppen durch OEM-Anwender ist eine leichte Anpassung an viele Einsatzforderungen realisierbar, wobei ein Systemausbau durch Einbindung spezieller Peripherie möglich ist.

Der AC A 7100 verfügt über alle Leistungs- und Strukturmerkmale der 16-Bit-Mikrorechnergeneration.

Die Leistungsfähigkeit des A 7100 wird nicht zuletzt von seiner Hauptspeicherkapazität (768 KByte), seiner Grafikfähigkeit sowie der Koppelfähigkeit zu übergeordneten Systemen bestimmt, die ihn auch für den Einsatz zur Lösung von CAD/CAM-Aufgaben befähigen.

Die CAD/CAM-Fähigkeit des AC A 7100 wird darüber hinaus unterstützt mit der Anschlußmöglichkeit entsprechender Peripherie aus dem Kombinat Robotron (Plotter K 6418, grafikfähige Drucker K 6313, K 6320, SD 1157 Typ 269, Digitalisiergeräte K 6401, K 6402, Grafik-Menütablett K 6405), mit der Bereitstellung einer Grafik-Grundsoftware (GSX-kompatibel) sowie einer flexiblen Firmware des Grafikkontrollers, der eine softwaremäßige Realisierung des Grafikkernsystems (GKS) unterstützt.

Der Arbeitsplatzcomputer A 7100 zeichnet sich u. a. mit folgenden Merkmalen aus:

- Neues, dem internationalen Trend entsprechendes Gestaltungskonzept, das eine bessere Anwenderfreundlichkeit in bezug auf die ergonomische Gestaltung, Geräuscharmut, Bedienung, Diagnose und Service realisiert.
- Neue Bildschirmtechnik, die eine leistungsfähige effektive Grafikverarbeitung erst ermöglicht.
- Deutlich verbesserte Leistungsfähigkeit gegenüber den vorhandenen Bürocomputern A 5120/5130 sowie PC 1715 durch neue Systemarchitektur C 16-Bit-Mikrorechner).
- Verbessertes Preis – Leistungsverhältnis
- Realisierung von Sonderfunktionen, die einen breiten Einsatz des Rechners in speziellen Anwendungsfällen ermöglichen
  - Möglichkeit des Ferneinschaltens
  - Lüftrausfallerkennung
  - Netzausfallerkennung
  - automtischer Systemanlauf bei Spannungszuschaltung
  - Laden verschiedener Betriebssysteme mit automatischer Parametererkennung

Fortsetzung Seite 5

Abb. 1: Arbeitsplatzcomputer A 7100

Tafel 1:
Allgemeine technische Daten des AC A 7100

1. Systembus
- logisch, elektrisch, funktionell
  - Systembus des Mikrorechner-Modulsystems 16
  - entspr. IEC 47 B (secretariat) 19
  - kompatibel zu MULTIBUS nach IEEE P 796
  - standardisiert im SRK als I 41
- konstruktiv
  - entspr. IEC 47 B (secretariat) 21 für Einsatz Doppeleuropaleiterplatten
  - steckerkompatibel zu AMS-Bus
  - standardisiert im SKR nach Normativmaterial 3.2 NM 12, Anlage 1
- funktionelle Merkmale
  - asynchroner Betrieb
  - 16 bit Datenbreite, 24 bit Adreßbreite
  - erlaubt Zusammenarbeit von 8- und 16-bit-Prozessoren
  - Interruptsystem; 8 Interruptleitungen, vektororganisiert
  - Busarbitragesystem, seriell oder parallel
  - gestattet Aufbau von Multiprozessorsystemen
- Bustaktfrequenz
  - 9,832 MHz

2. Zentrale Verarbeitungseinheit (ZVE)
- Mikroprozessor
  - K 1810 WM 86 5 MHz
  - Verarbeitungsbreite 8, 16 bit
  - Befehlswortlänge 1–6 Bytes
  - Befehlssystem 70 Basisbefehle
  - Adressierungsarten 30
  - Registersatz
  4 universelle Register (16 bit)
  2 Pointerregister (16 bit)
  2 Indexregister (16 bit)
  4 Segmentregister (16 bit)
  Befehlszähler
  Statusregister
  - coprozessorfähig durch LOCK-Mechanismus
  - 40 pin DIL, eine Spannung + 5 V
- Befehlsausführungszeiten (Register – Register)
  0,4 µs aus CPU-Cache
  1,2 µs mit Zweitspeicher
  1,4 µs mit Operativspeicher
- Speicheradreßraum 1 MByte
- E/A-Adreßraum 64 KByte
- CPU-Taktfrequenz 4,915 MHz
- Interruptsystem
  - vektororganisierte Interruptbehandlung
  - 1 NMI (nicht maskierbar)
  - 8 INTR (maskierbar)
  - erweiterungsfähig bis auf 64 Interruptebenen
  - 18 Interruptquellen, wählbar über Interrupmatrix
- Stackorganisation vorhanden
- Timeoutüberwachung ca.10 ms, abschaltbar
- WAIT-Steuerung vorhanden, extern nutzbar
- programmierbarer Intervalltimer KR 580 WI 53, verschiedene einstellbare Eingangsfrequenzen
- Interfaces auf ZVE
  IFSS, IFSP-M (Centronics)

3. Speicher
- Operativspeicher (RAM)
  - 256...768 K Byte
  - auf Basis Operativspeicher-Modul mit 256 KByte
  - Zugriffszeit 560 ns
  - Schaltkreisbasis KU 565 RU 56, 64 K×1 Bit
  - Zugriff wort- oder byteweise
  - Anfangsadresse in Stufen von 128 KByte im Adreßraum 0...1 MByte einstellbar
  - Datensicherung über byteweise Paritätskontrolle
  - Refresh dezentral
- Festwertspeicher (EPROM)
  - max. 32 KByte auf Zentraler Verarbeitungseinheit
- Externspeicher
  - Folienspeicher
  - 1 MByte in Rechnergrundgerät (2×K 5600.20)
  - 1 MByte zusätzlich in Beistellgefäß K 5672 (2×K 5600.20) oder 0,8 MByte zusätzlich in Beistellgefäß K 5671 (2×K 5600.10)
  - Rechnergrundgerät ist konstruktiv für Einbau eines 5¼"-Festplattenspeichers (10 MByte) vorbereitet
  - Steuerung über E/A-Prozessor, DMA-Verkehr am Systembus
  - max. Übertragungsrate 500 KByte/s
  - Kommunikation mit ZVE analog ISBC 215A/218

4. Interfaces
4.1. IFSS
- Interface-Typ
  - 20 mA Sende- und Empfangsschleife Duplex, seriell,
  „0" = 0...3 mA
  „1" = 15...25 mA
- Übertragungsart
  - asynchron
- Arbeitsmodi
  - aktiv oder passiv, für jede Stromschleife einstellbar
- Übertragungsrate
  - max. 9600 Band, programmierbar
- Zeichenformat
  - programmierbar
  1 Startbit, 5/6/7/8 Datenbits, 1 Paritätsbit, 1/1,5/2 Stoppbits
- Break-Erkennung
  - vorhanden
- Schutzgrad
  - Zusatzisolation nach IEC 380/435 im Passivmodus
  - Galvanische Trennung über Optokoppler im Empfänger
- Interface-Steckverbinder
  - D-Subminiatur, 25-polig, EBS-GO 4006/01 Buchseleiste auf Modul
- Übertragungsentfernung
  - max. 500 m bei 9600 Band
  - größere Übertragungsentfernung bei verringerter Bandrate
- Interface-Kabel
  - geschirmtes Rundkabel, 2 verdrillte Paare, max. 200 Ohm je Leitungspaar
- Sonstiges
  - optische Anzeige der Betriebsbereitschaft

4.2. S2 (V24)
- Interface-Typ
  - Duplex, seriell, pegelgesteuerte Signale
  Signalpegel: EIN + 3V... + 12V
  AUS – 3V... – 12V
- Übertragungsart
  - synchron, asynchron auf Standleitung oder vermittelten Zwei- oder Vierdraht-Übertragungswegen
- Übertragungsrate
  - asynchron: max. 9600 Baud
  synchron: max. 20 K Baud
- Übertragungsentfernung
  - 15 m (bis Modem-, DÜE-Anschluß)
- Interface-Kabel
  - unsymmetrische Leitung, geschirmtes Rundkabel

- Interface-Steckverbinder
  – D-Subminiatur, 25-polig, EBS-GO 4006/01 Buchsenleiste auf Modul
- Betriebsmodi
  – Simplex, Halbduplex, Düplex
- Sonstiges
  – optische Anzeige der Betriebsbereitschaft

4.3. IFSP

- Inteface-Typ
  – byte-parallel, Ausgabe, TTL-Pegel (40 mA)
- Übertragungsart
  – Abfrage-Betrieb, synchrone/asynchrone Statusübertragung
- Übertragungsrate
  – max. 20 KByte/s
- Übertragungsentfernung
  – max. 15 m
- Interface-Kabel
  – geschirmtes Rundkabel, Z = 100 ± 20 Ohm, abgeschlossen
- Interface-Steckverbinder
  – D-Subminiatur, 25polig, EBS-GO 4006/01 Steckerleiste auf Modul
- Sonstiges
  – optische Anzeige der Betriebsbereitschaft

4.4. IFSP-M (Centronics)

- Interface-Typ
  – bit-parallel, byte-seriell, Ausgabe
- Logische Bedingungen
  – TTL-Pegel: Low ≤ 0,4 V
    High ≥ 2,4 V
    Ausgangsstrom
    Low I_{OL} = 32 mA bei U_{OL} = 0,4 V
    Ausgangsstrom
    High $-I_{OH}$ = 5 mA bei U_{OH} = 2,4 V
- Übertragungsrate max. 2,2 KByte/s
- Übertragungsentfernung
  – max. 3 m
- Interface-Kabel
  – geschirmtes Rundkabel, paarweise verdrillte Leitungen, Adernquerschnitt min. 0.08 mm²
  Wellenwiderstand 60...150 Ohm
  Leitungsabschluß 3,3 kOhm nach +5 V
- Interface-Steckverbinder
  – D-Subminiatur, 25polig, EBS-GO 4006/01 Steckerleiste auf Modul
- Zeitbedingungen
  – Datenverbreitungszeit 1 µs min
  Datenhaltezeit 1 µs min
  STROBE-Impulsbreite 1 µs min
  ACK-Impulsbreite 5µs max

5. Anschließbare Geräte

- Drucker K 6311/12/13/14/20
  SD 1152, SD 1157
- Grafik-Menütablett K 6405
- Plotter K 6418
- Digitalisiergeräte K 6401, K 6402

6. Betriebssysteme

- SCP 1700
  – Einzelnutzersystem
- BOS 1810
  – echtzeitfähiges Multiprogrammsystem
- MUTOS 1700
  – universelles Timesharing-System

7. Energieversorgung

– Wechselspannung
220 V +10 % – 15 %, 47–63 Hz
– Wirkleistung 360 W bei sekundärseitiger Auslastung der Stromversorgungsmoduln
– Stromaufnahme max. 2,5 A

8. Umgebungsbedingungen

– Einsatzklasse EK 3, Tab. 1 nach
– Transportklasse TK 2, Tab. 2 TGL
– Lagerklasse 2, Tab. 7 26465

9. Sicherheitsstatus

– nach IEC 380/VDE 0806 mit Schutzklasse I
– Sicherheitskleinspannung
– Schutzgrad IP 20 nach TGL RGW 778

10. Funkentstörung

– entspr. Grenzwertklasse B nach VDE 0871
– Einhaltung der Grenzwerte für Funkentstörungen F1 bis F4/12 nach TGL 20885/12

11. Ferneinschaltung

– automatischer Anlauf wie bei Netzeinschaltung, gesteuert über Signal Nr. 125 des Interface S2 (V24)
– programmierte Einstellung der Bereitschaft
– Zustandsanzeige durch grünes Dauerlicht

12. Lüfterüberwachung

– optische Drehzahlüberwachung durch Miniatur-Reflexkoppler
– automatisches Abschalten des Rechners bei Unterschreitung der Drehzahl von 900 ± 200 U/min
– Zustandsanzeige durch Blinken der gelben Bereitsschafts-Anzeige

13. Konstruktion

- Ausführung
  – alle Systemkomponenten als Auftischgeräte
- Abmessungen
  – Rechnergrundgerät (HxBxT):
  174 mm × 486 mm × 457 mm
  Bildschirmeinheit (HxBxT):
  290 mm × 330 mm × 360 mm
  Tastatur (HxBxT):
  60 mm × 461 mm × 239 mm
  Beistellgefäß K 5671 (HxBxT):
  250 mm × 275 mm × 535 mm
  Beistellgefäß K 5672 (HxBxT):
  174 mm × 225 mm × 485 mm
- Module
  – Doppeleuropaleiterplatten, MLL, 233, 4 × 160 mm, Gernischtraster
  Steckraster entspr. Bauhöhe: 20,32 mm oder 15,24 mm
- Steckverbinder
  – System- und Nebenbus: 96polig, IEC 603-2,
  C6M-C1A DIN 41612 Steckerleiste auf Modul
  C6F-C1H DIN 41612 Buchsenleiste auf Rückverdrahtung
  – Interfaces
  D-Subminiatur-Anschlüsse
  EBS-GO 4006, voll geschirmt, 9- und 25-polig, meistens direkt auf Modul
  – Geräteinterne Verbindungen:
  EFS-Steckverbinder, Schlitzklemmtechnik, TGL 37912

Fortsetzung von Seite 2

- Flexible Firmware mit leistungsfähigem Monitor (analog 957B) einschließlich System-Confidence-Test.
- Umfangreiche Prüfsoftware zur Realisierung eines verbesserten Fehlerdiagnose- und Kundendienstkonzeptes.

Tafel 1 gibt einen Überblick über die allgemeinen technischen Daten des AC A 7100.

2. Systemkomponenten und konstruktiver Aufbau des AC A 7100

2.1. Systemkomponenten

Der Arbeitsplatzcomputer A 7100 ist ein Auftischgerät, das in seiner Grundkonfiguration (Abb. 1) aus folgenden getrennt aufstellbaren Systemkomponenten besteht (Tafel 2):

- Rechnergrundgerät (RGG) K 1710
- Bildschirmeinheit K 7229.22
- Tastatur K 7637.9X

Die Grundkonfiguration des AC A 7100 kann durch Anschluß peripherer Systemkomponenten, ebenfalls in Auftischausführung, erweitert werden (Tafel 3):

- Folienspeichereinheit K 5671 mit zwei Laufwerken K 5600.10 (8")
- Minifolienspeichereinheit K 5672 mit zwei Laufwerken K 5600.20 (5,25")
- Hardcopy-Drucker K 6311 oder K 6312
- Grafik-Menütablett K 6405

Darüber hinaus ist über die im AC A 7100 realisierten Standardinterfaces (siehe Tafel 1) weitere einsatzspezifische Peripherie aus dem Kombinat Robotron anschließbar.

Tafel 2:
Systemgrundkomponenten des AC A 7100

1. Rechnergrundgerät K 1710

- Abmessungen (H x B x T)
  – 174 mm × 486 mm × 451 mm
- Baugruppen
  – Verdrahtungsbaugruppe
  – Modulaufnahme, 10 Steckeinheitenplätze,
  Steckraster 20,32 mm und 15,24 mm
  – Frontbaugruppe mit Bedienfeld
  – Stromversorgungsmodule STM 5 V/40 A, STM 5 V/100 mA, STM + 12 V/4,0 A, –12 V/0,5 A
  – Speicherbaugruppe,,2x MFSE K 5600.20
  – Netzeingang
  – Lüfter, Lüfterüberwachung
- Logikmodule
  – Mehrlagenleiterplatten (MLL) 233,4 × 610 mm² (Doppeleuropaformat)
- Busssteckverbinder
  – 96polig, C6F-C1H DIN 41612 (Buchsenleiste)
- Interface-Steckverbinder
  – D-Subminiatur EBS-GO 4006, 9- und 25polig, voll geschirmt, meistens direkt auf Logikmodul

2. Bildschirmeinheit K 7229.22

- Abmessungen (H x B x T)
  – 290 mm × 330 mm × 360 mm
- Drehbarkeit/Neigbarkeit
  – 45°/15°
- Bildschirmdiagonale 310 mm
- Bildwechselfrequenz 50–60 Hz
- Horizontalablenkung ca. 22 KHz
- Bildschirm
  – monochromatisch, grün, reflexgemindert
- Helligkeitsregelung
  – stufenlos durch Bediener programmierbar in 3 Stufen
- Energieversorgung
  220 V + 10 %– 15 %, 47–63 Hz
- Wirkleistung ca. 35 W
- Sicherheitskleinspannung
  – nach STW RGW 3743-82/TGL 42340
- Schutzgrad
  – IP 20 nach TGL 15165
- Video-Interface
  – X4, Subminiatur-D, 9polig, Buchsenleiste auf Modul

3. Tastatur

- Abmessungen (H x B x T)
  – 60 mm × 461 mm × 239 mm
- Varianten
  – K 7637.91 lateinisch/deutsch K 7637.92 lateinisch/kyrillisch
- Ausgabe
  – 8-bit-Zeichen als Einzelzeichen und Zeichenfolge
- Zeichensatz
  – 2 umschaltbare Zeichensätze nach TGL RGW 356-76 (KOI-7), Unterscheidung durch 8. Bit
- Tastensatz
  – 106 Tasten, wartungsfrei, Tastenhub 4 mm, durch Umschaltung jede Taste mit bis zu 5 Codes belegt
- Diagnosemittel
  – automatischer Selbsttest, über Zentrale Verarbeitungseinheit abfragbar
- Anzeigen
  – 6 LED-Anzeigen
  Akustischer Geber
- Energieversorgung
  – +5 V ±5 %, 500 mA, Versorgung vom Rechnergrundgerät über Interfacekabel
- Schutzeigenschaften
  – Sicherheitskleinspannung nach IEC 380
  – Interface
  IFFS, Duplex, Sender aktiv
  9600 Baud, Format: 1 Startbit, 8 Datenbits, 2 Stoppbits; Steckverbinder Subminiatur-D, 9polig, Steckerleiste am RGG

2.2. Konstruktiver Aufbau

Das Rechnergrundgerät (RGG) K 1710 des AC A 7100 mit den Abmessungen (H x B x T) 174 mm × 486 mm × 451 mm ist ein Auftischgerät und enthält folgende Baugruppen (Abb. 2):

- Verdrahtungsbaugruppe (1)

Sie dient zur elektrischen Verbindung der Logikmodule über zwei Rückverdrahtungsleiterplatten (9), zur Verteilung von Netz- und Kleinspannungen sowie Logiksignalen und ist in der Mitte des Rechnergrundgerätes angeordnet, so daß die Logik- und Stromversorgungsmodule von vorn und hinten gesteckt und damit elektrisch verbunden werden. Die übrigen Baugruppen sind ebenfalls steckbar mit der Verdrahtungsbaugruppe verbunden. Auf den Rückverdrahtungsleiterplatten (Mehrlagen-Leiterplatte mit den Abmessungen 130 mm × 100 mm) kommen 96polige indirekte Steckverbinder nach IEC 603.2 Typ C6-F-C1H DIN 4112 (Federleiste) zum Einsatz.

- Kassette für Aufnahme der Logikmodule (2)

Im linken hinteren Teil des Rechnergrundgefäßes befindet sich eine Kassette mit 7 Steckplätzen zur Aufnahme der Logikmoduln.

Drei weitere Steckplätze stehen im vor-

Tafel 3: Periphere Systemkomponenten des AC A 7100

1. Folienspeichereinheit FSE K 5671
- Abmessungen (H x B x T)
  – 250 mm × 275 mm × 555 mm
- Anzahl der Laufwerke
  – 2 (8"), K 5600.10
- Abmessungen der Diskette
  – 203,2 mm × 203,2 mm
- Speicherkapazität je Diskette
  – 250 KByte
- Anzahl der Köpfe 1
- Spurdichte 48 tpi
- Anzahl der Spuren 77
- Übertragungsrate 250 KBit/s
- Aufzeichnungsverfahren FM
- Energieversorgung
  220 V + 10 % – 15 %, 47–63 Hz
- Wirkleistung 140 W
- Interface
  – Linieninterface, Anschluß an Modul AFS

2. Minifolienspeichereinheit MFSE K 5672
- Abmessungen (H x B x T)
  – 147 mm × 225 mm × 405 mm
- Anzahl der Laufwerke
  – 2 (5,25"), K 5600.20
- Abmessungen der Diskette
  – 133,3 mm × 133,3 mm
- Speicherkapazität je Diskette
  – 400 KByte
- Anzahl der Köpfe 1
- Spurdichte 96 tpi
- Anzahl der Spuren 80
- Übertragungsrate 250 KBit/s
- Aufzeichnungsverfahren FM, MFM
- Energieversorgung
  220 V + 10 % – 15 %, 47–63 Hz
- Wirkleistung 70 W
- Interface
  – Linieninterface, Anschluß an Modul AFS

3. Hardcopy-Drucker K 6311/12
- Abmessungen (H x B x T)
  – K 6311 130 mm × 370 mm × 280 mm
  K 6312 130 mm × 520 mm × 280 mm
- Masse
  – K 6311 ca. 6 kg,
  K 6312 ca. 9 kg
- Druckprinzip
  – Matrixdrucker, mit Nadeldrucksystem, seriell
- Druckgeschwindigkeit: 100 Zeichen/s
- Druckrichtung
  – vorwärts und rückwärts mit Druckwegoptimierung
- Zeichenumfang
  – 95 Zeichen, nationale Zeichensätze
- Zeichencode
  – ISO/CCITT Nr. 5 ASCII
- Steuercodes
  – Steuerzeichen sowie ESC-Folgen
- Formate (Zeichen/Zeile)

	K 6311	K 6312
Zeichen/Zeile abhängig von gewählter Zeichendichte	80–120	132–198
Papierbreite mm		
Einzelblatt oder Rolle	216	375
Leporello	265	410

- Farbträger Farbbandkassette
- Geräuschpegel unter 60 dB
- Interfaces
  – Parallel: IFSP, Centronics
  Seriell: IFSS, V24
- Energieversorgung
  – 220 V + 10 %, –15 %, 47–63 Hz
- Wirkleistung ca. 50 W

4. Menü-Tablett K 6405 (vorläufige Angaben)
- Abmessungen (H x B x T)
  – 35 mm × 490 mm × 410 mm
- Masse ca. 5 kg
- Aktive Arbeitsfläche
  – ca. 297 mm × 210 mm
- Auflösung 0,1 mm
- Genauigkeit 0,5 mm
- Abtastrate min. 50/s
- Meßwertaufnehmer
  – Stift, Kursor optional
- Grundbetriebsarten
  – POINT, RUN, TRACK, HALT
- Energieversorgung
  – +5 V, +12V, –12 V insgesamt ca. 25 W über RGG
- Interface
  – V24 Nahanschluß, 19,2 K Baud
  Anschluß an Modul KGS

deren Teil unterhalb der Speicherbaugruppe zur Verfügung.
- Frontbaugruppe (3)

Die Frontbaugruppe (FGB) enthält neben der Logik für die Lüfterüberwachung, Netzausfallerkennung, akustischen Geber und Ferneinschaltung ein minimales Bedienfeld mit drei Anzeigen (RUN, HALT, PWRON/REMOTE) und einer RESET-Taste.
- Stromversorgungsmodule (4)

Zwei Stromversorgungsmodule sind auf der rechten Seite des Rechnergrundgerätes im vorderen und hinteren Teil montiert. Neben dem vorderen Modul befindet sich ein kleiner Hilfsspannungsmodul zur Speisung der Frontbaugruppe.
- Speicherbaugruppe (5)

Die Speicherbaugruppe besteht aus zwei nebeneinander montierten Minifolienspeicherlaufwerken K 6500.20, die über Führungsschienen in das Rechnergrundgerät eingeschoben und mittels eines Spannmechanismus leicht lösbar arretiert werden. Interface und Stromversorgung werden über Flachbandkabel bzw. Rundkabel steckbar zugeführt.
- Netzeingang (6)

Diese Baugruppe nimmt Netzstecker, -schalter, -sicherungen und das Filter auf. Über Kabel wird die Netzspannung steckbar an die Verdrahtungsbaugruppe geführt.
- Lüfter (7)

Das Rechnergrundgerät ist mit zwei Lüftern ausgestattet, die das Gerät von links nach rechts belüften. Zur Funktionsüberwachung werden Reflexkoppler eingesetzt. Zur Luftfilterung befindet sich im linken Seitenteil eine Filtermatte.

Alle Baugruppen sind auf einem wannenförmigen Bodenteil montiert, so daß sich ein kompakter Geräteaufbau ergibt, der durch leicht lösbare Verkleidungsteile (zwei Seitenteile, Deckel, Frontblende, Rückwand) komplettiert wird.

Das Rechnergrundgerät ist konstruktiv und elektrisch für den Einbau eines 5¼"-Festplattenspeichers (10 MByte) anstelle eines Minifoliespeicherlaufwerkes vorbereitet. Die Interfaceanschlüsse des Rechners werden direkt

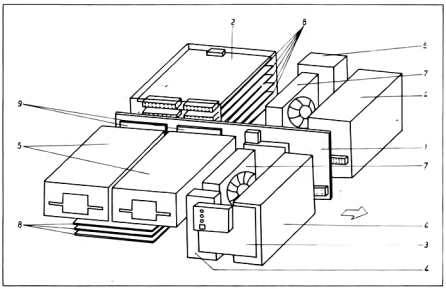


Abb. 2: Konstruktion Rechnergrundgerät K 1710 des AC A7100

von der Griffseite der entsprechenden Logikmodule durch die Rückwand des Gerätes herausgeführt.
Tastatur- und Menü-Tablett-Anschluß befinden sich auf der linken Seite des Rechnergrundgerätes.
Die Bildschirmeinheit mit eigener Stromversorgung wird über das Video-Interface an das Rechnergrundgerät angeschlossen. Sie ist absetzbar, dreh- und schwenkbar und wird gewöhnlich auf das Grundgerät gestellt.
Die Tastatur ist frei beweglich und wird über ein Serielles Interface (IFSS) angeschlossen, wobei die Stromversorgung über das Interfacekabel durch das Grundgerät erfolgt.

2.3.
Stromversorgung des AC A 7100

Das Rechnergrundgerät K 1710 ist mit folgenden Stromversorgungsbaugruppen bestückt (Tafel 4):

- Stromversorgungsmodul
  STM 5 V/10 A

Dieser Stromversorgungsmodul ist als Schaltnetzteil nach dem Flußwandlerprinzip für universelle Anwendbarkeit ausgelegt und liefert eine 5-V-Ausgangsspannung, die mit max. 40 A belastbar ist. Der Modul ist mit Fernfühleranschlüssen ausgerüstet und kann durch Anlegen einer Kleinspannung abgeschaltet werden. Ein Spannungsausfallsignal kündigt bei einer Stützzeit von min. 10 ms einen bevorstehenden Zusammenbruch der Ausgangsspannung an.
Der Modul ist mit Schutzschaltungen wie Überspannungsschutz, Überstrombegrenzung und Unterspannungsabschaltung versehen.

- Stromversorgungsmodul
  STM + 12 V/4 A; – 12 V/0,5 A

Diese Stromversorgungsbaugruppe ist als Schaltnetzteil nach dem Sperrwandlerprinzip speziell für das Rechnergrundgerät K 1710 ausgelegt und liefert zwei Ausgangsspannungen +12 V/–12 V, die jeweils mit 4 A bzw. 0,5 A belastbar sind. Der Modul ist mit äquivalenten Schutzschaltungen wie der STM 5 V/40 A versehen und ebenfalls gesteuert abschaltbar. Ein Spannungsausfallsignal wird bereitgestellt.

- Stromversorgungsmodul
  STM + 5 V/0,1 A

Diese Stromversorgungsbaugruppe dient als Hilfsspannungsmodul zur getrennten Stromversorgung der Frontbaugruppe und zur Realisierung der Ferneinschaltfunktion. Sie ist konventionell mit Transformator aufgebaut.
Die Funktion der Stromversorgungsbaugruppen ist bei einer eingangsseitigen Spannung von 187 V...242 V, 47 Hz...63 Hz gewährleistet. Sie sind in der Schutzklasse I ausgeführt und bieten für alle Ausgangsspannungen den Status „Sicherheitskleinspannung".
Die Bildschirmeinheit K 7229 sowie die Beistellgeräte für die Foliespeichereinheiten K 5671 und K 5672 besitzen einen eigenen Netzanschluß mit Schalter, Sicherung, Anzeige und Netzfilter. Es kommen Stromversorgungsmodule aus der bekannten DEKK-Reihe zum Einsatz.
Die Tastatur K 7637.9x und das Grafik-Menütablett K 6405 verfügen über keinen eigenen Netzanschluß. Sie werden von dem Rechnergrundgerät K 1710 gleichspannungsmäßig über die Interfacekabel gespeist.

2.4.
Modulbestand

Der Arbeitsplatzcomputer robotron A 7100 kann durch Ausrüstung mit verschiedenen Logikmoduln den speziellen Einsatzfällen optimal angepaßt werden. Darüber hinaus besteht die Möglichkeit des Einsatzes spezieller Baugruppen aus dem Mikrorechner-Modulsystem 16.
Zum Modulbestand des AC A 7100 gehören die in Tafel 5 angegebenen Logikmodule. Jeder Logikmodul bildet eine abgeschlossene Funktionseinheit, die auf einer Mehrlagenleiterplatte (MLL) mit vorrangig 4 Ebenen in Gemischtraster (Zollmetrisch) im Format 233,4 mm × 160 mm (Doppeleuropaformat) realisiert ist. Auf der Systembusseite sind je nach Funktionsumfang des Moduls 1 oder 2 96polige indirekte Steckverbinder vom Typ C6M-C1A DIN 41612 (Messerleiste) angeordnet. Auf der Griffseite befinden sich die Interfaceanschlüsse (wenn vorhanden), für die das Subminiatur-D-Steckverbindersortiment EBS-GO 4006 mit unterschiedlichen Kontaktzahlen (9- und 25polig) in voll geschirmter Ausführung verwendet wird sowie eine Frontblende mit zwei Griffelementen und zwei unverlierbare Schrauben zur Befestigung des Moduls im Rechnergrundgerät.
Abhängig vom Bauelementeeinsatz gibt es Logikmodule mit den Bauhöhen 13,5 mm oder 8,5 mm, die im Steckraster von 20,32 mm bzw. 15,24 mm einsetzbar sind. Die Einbaulage der Module ist horizontal.
Eine ausführliche Funktionsbeschreibung der einzelnen Logikmodule ist in /1/ dargelegt.

3.
Systemarchitektur des Arbeitsplatzcomputers A 7100

Die Systemarchitektur des Arbeitsplatzcomputers A 7100 wird maßgeblich durch das verwendete 16-Bit-Mikropro-

zessor-Schaltkreissystem K 1810 sowie durch den standardisierten Mikrorechnerbus (Systembus MMS 16) geprägt. Sie basiert auf der Systemphilosophie des Mikrorechner-Modulsystems MMS 16, die auf die grundsätzlich durchgängige modulare Gestaltung orientiert. Durch Auf- und Abrüstbarkeit, Austausch- und Erweiterbarkeit von einzelnen Baugruppen (Module) entsprechend dem jeweiligen technisch-technologischen Stand können Systemstrukturen in einem weiteren Leistungsbereich für die verschiedensten Anwendungsfälle einheitlich konfiguriert werden. Diese Konzeption gewährleistet eine Systemoffenheit und gestattet einen universellen Einsatz, den Ausbau, die Steigerung der Leistungsfähigkeit und somit die Nutzung des Systems über mehrere Rechnergenerationen.

Die Systemarchitektur des AC A 7100 (Abb. 3) wird durch folgende Merkmale charakterisiert:

- Standardisierter multimasterfähiger Systembus
- Einsatz intelligenter Subsysteme zur Funktionsteilung und Entlastung des Systembus
- Verwendung von lokalen Nebenbussen innerhalb der Subsysteme
- Systemoffenheit
- Verwendung international üblicher Standardinterfaces

Die Systemarchitektur des AC A 7100 läßt ein hierarchisch aufgebautes funktionsgeteiltes Multiprozessorsystem erkennen, in dem mehrere Subsysteme mit fest definierten Funktionen zum Einsatz kommen. Die Kommunikation der Subsysteme mit der System-ZVE K 2771 erfolgt nach dem Auftragsprinzip mit fest vereinbarten – aber für die einzelnen Subsysteme unterschiedlichen Regeln. Diese Funktionsteilung wirkt sich äußerst günstig auf die Systemeigenschaften aus:

- Da alle Subsysteme mit eigener Intelligenz (d. h. mit eigenen Prozessoren) ausgerüstet sind, wird die Parallelarbeit von System-ZVE und Subsystem möglich, wodurch eine Erhöhung des Befehlsdurchsatzes erreicht wird.
- Verringerung der Systembusbelastung
- Verbesserung des Echtzeitverhaltens der System-ZVE durch Entlastung von E/A-Operationen
- Realisierung einer Modularität und damit Einsatz in verschiedenen Rechnergenerationen
- Vereinfachung der E/A-Programmierung.

Die Erweiterungsfähigkeit des AC A 7100 wird durch den Einsatz international üblicher Steckverbindersysteme und Standardinterfaces unterstützt.

Tafel 5:
Logikmodule des AC A 7100

Modultyp	Chiffre	Funktion	Bus-Steckverbinder X1	Bus-Steckverbinder X2	arbeitet am Systembus als
ZVE	K 2772	Zentrale Verarbeitungseinheit	+	+	Master
ZPS	K 2071	Zweiportspeicher	+	+	Slave
OPS	K 3571	Operativspeicher	+	–	Slave
KES	K 5170	Kontroller für Externsp.	+	+	Master
AFS	K 5171	Anschlußsteuerung für Festplattenspeicher	–	+	–
AFP[1]	K 5172	Anschlußsteuerung für Festplattenspeicher	–	+	–
KGS	K 7070	Kontroller für Grafik-Subsystem	+	+	Slave
ABS[2]	K 7071	Alphanumerische Bildschirmansteuerung	+	–	Slave
ABG	K 7072	Anschlußsteuerung für Bildschirm, grafisch	+	+	–
ASP	K 8071	Anschlußsteuerung für serielle und parallele Interfaces	+	–	Slave

[1] in Vorbereitung
[2] alternativ zu KGS/ABG einsetzbar

Tafel 4:
Stromversorgungsmodule AC A 7100

Einsatz	Universal-Module	Spezial-Module
Rechnergrundgerät K 1710	+5 V/200 W	+12 V, –12 V/100 W +5 V/0,1 A Hilfsspannung
Beistellgefäß (8") K 5671	+5 V/50 W +12 V/50 W	
Beistellgefäß (5,25") K 5672		+5 V, + 12V/50 W
Wirkprinzip	Flußwandler	Sperrwandler

3.1. Logikstruktur des AC A 7100

Abb. 4 zeigt die im Arbeitsplatzcomputer A 7100 realisierte Logikstruktur auf Basis der in Tafel 5 angegebenen Logikmodule. Alle an den Systembus (über Steckverbinder X1) anschließbaren Module bilden in der Regel abgeschlossene Funktionseinheiten, die als Master oder Slave arbeiten. Die Ausrüstung mit verschiedenen Funktionseinheiten kann variiert werden, ohne die Funktions- und Kommunikationsfähigkeit der verbleibenden Einheiten zu stören, wodurch sich der weitgehend modulare Systemaufbau ergibt. Die intelligenten Subsysteme lassen sich auf einem Modul realisieren (z. B. ABS K 7071) oder nutzen bei größerem Funktionsumfang die Verbindung über einen Nebenbus (z. B. Grafik-Subsystem). Intelligente Subsysteme mit großem Datendurchsatz arbeiten als Master am Systembus (z. B. E/A-Subsystem), Subsysteme mit geringem Datendurchsatz und starker Abhängigkeit von der System-ZVE arbeiten als Slave (z. B. Grafik-Subsystem).

3.2. Bussystem des AC A 7100

Die Logikstruktur des Arbeitsplatzcomputers A 7100 nach Abb. 4 läßt aus logischer Sicht drei unabhängige Buskomplexe erkennen:

- Systembus
- ZPS-Nebenbus
- lokale Nebenbusse für Subsysteme

3.2.1. Systembus

Als universeller Systembus wird der international übliche Mikrorechnerbus verwendet, der im SKR als I 41-Bus und im IEC sowie DIN standardisiert ist. Er entspricht logisch-funktionell und elektrisch dem MULTIBUS entsprechend IEC 47 B (Secretariat) 19 sowie konstruktiv dem AMS-Bus entsprechend IEC 47 B (Secretariat) 21 für den Einsatz von Doppeleuropaleiterplatten. Der Systembus wird durch folgende Merkmale charakterisiert:

– Multiprozessorfähigkeit

Der Verkehr auf dem Bus geschieht nach dem bekannten Master-Slave-Prinzip. Dabei ist nicht nur der Anschluß mehrerer Slaves in üblicher Weise, sondern auch der Anschluß mehrerer Master erlaubt.

– Schnelles Busmasterwechselsystem

Der Bus besitzt entsprechende Steuersignale, die es erlauben, die Busmaster entweder in einer seriellen oder einer parallelen Arbitrage-Schaltung zu verbinden und ihnen somit prioritätsabhängig das Zugriffsrecht zu den Busressourcen zu gestatten.

– großer Speicheradreßraum (16 MByte)

Der ZVE-Modul K 2771 des AC A 7100 bedient 20 Adreßleitungen und bietet somit nur einen Speicheradreßraum von 1 MByte.

– getrennter Ein-/Ausgabe-Adreßraum (64 KByte)

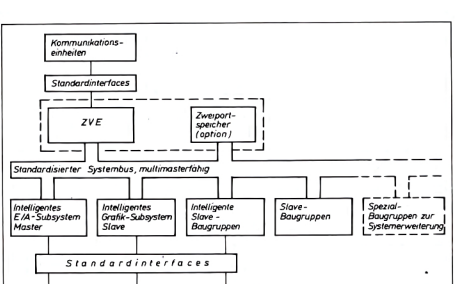
– 16 Bit breiter Datenbus mit voller Anschlußkompatibilität für 8- und 16-Bit-Baugruppen durch Realisierung eines Swapping-Mechanismus.

– flexibles, erweiterungsfähiges, vektororganisiertes Interruptsystem mit 8 separaten Interrupt-Busleitungen.

– Sonderfunktionen zur Realisierung von Adreßbereichsüberlagerungen und zur Unterstützung der Arbeit mit Zweiportspeichern durch INHIBIT- und

– asynchrones Signalspiel mit Quittungssignal zur Geschwindigkeitsanpassung.

– hohe Bustaktfrequenz (10 MHz).



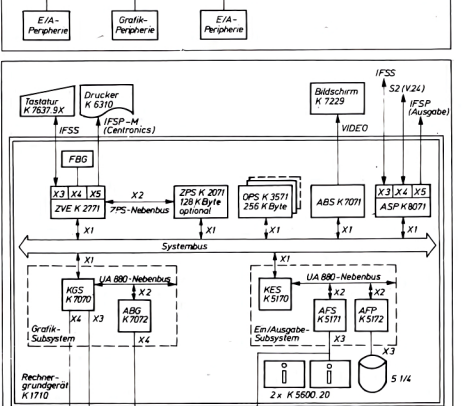


Abb. 4: Struktur des AC 7100

3.2.2. ZPS-Nebenbus

Der ZPS-Nebenbus ist aus logischer Sicht ein spezieller lokaler Nebenbus und dient im AC A 7100 zur Verbindung der Zentralen Verarbeitungseinheit ZVE K 2771 mit dem Zweiportspeicher ZPS K 2071. Dabei ist die ZVE allein ohne ZPS voll arbeitsfähig, während der ZPS auf die Zusammenarbeit mit der ZVE angewiesen ist und somit einen optionalen Zusatz bildet, der beim Aufbau von leistungsfähigen Multiprozessorsystemen notwendig wird. Der ZPS-Nebenbus ist isomorph zum Systembus und kann bei Einsatz der ZVE K 2771 in OEM-Systemen zum Anschluß weiterer lokaler Ressourcen verwendet werden.

3.2.3. Lokale Nebenbusse für Subsysteme

Die lokalen Nebenbusse innerhalb der im AC A 7100 zum Einsatz gelangten intelligenten Subsysteme dienen zur Interfacebildung zwecks Anschluß und Erweiterung der lokalen Ressourcen.
Aus logischer Sicht lassen sich folgende lokale Nebenbusse unterscheiden:
– lokaler Nebenbus für E/A-Subsystem
– lokaler Nebenbus für Grafik-Subsystem.
Beide lokalen Nebenbusse sind UA-880-spezifische Busse.

3.2.4. Physische Realisierung des Bussystems

Die physische Realisierung des Bussystems des AC A 7100 erfolgt auf der Verdrahtungsbaugruppe durch zwei getrennte nebeneinanderliegende Rück-verdrahtungsleiterplatten (4 Ebenen-Mehrlagenleiterplatte, 230 mm × 100 mm) mit den Steckverbinderreihen X1 und X2. Als Steckverbinder kommen 96polige Buchsenleisten vom Typ C6F-C1H DIN 41612 zum Einsatz. Die Steckverbinderreihe X1 realisiert den Systembus (SB) und ist durchgängig auf den Steckplätzen 1 bis 7 vorhanden. Die Steckverbinderreihe X2 ist unterteilt und bildet so zwei Subsystembusse (SSB 1 und SSB 2), die jeweils einige direkt aufeinanderfolgende Steckplätze verbinden. Zwecks optimaler Raumausnutzung und kurzer Leitungslängen sind die Steckverbinder X2 von beiden Seiten auf der Rückverdrahtungsleiterplatte bestückt, so daß die Logikmoduln teils von der Rückseite des Rechnergrundgerätes her, teils von der Vorderseite kontaktiert werden. Zwischen den Steckverbindern sind Busbedämpfungswiderstände auf den Rückverdrahtungsleiterplatten untergebracht. Beim Einsatz von Logikmoduln ist die fest vorgegebene Prioritätsverteilung der Steckplätze zu berücksichtigen und darauf zu achten, daß die Prioritätskette entsprechend der realisierten Konfiguration geschlossen ist (Tafel 6).

Tafel 6: Physische Realisierung der Busstruktur im AC A 7100

Anordnung auf RVLP	Steckplatz-Nr.	Busstruktur		Prioritätsverteilung			Steckraster mm
		Stv. X1	Stv. X2	SB	SSB1	SSB2	
zur RGG-Rückseite	7			4	2		20,32
	6			2	1		15,24
	5			1		1	20,32
	4			5		4	20,32
	3			3			20,32
	2			6			20,32
	1			7			20,32
zur RGG-Vorderseite	10	–				2	15,24
	9	–				3	15,24
	8	–				5	15,24

SB – Systembus
SSB 1 – Subsystembus 1
SSB 2 – Subsystembus 2
– – kein Busanschluß
RGG – Rechnergrundgerät
RVLP – Rückverdrahtungsleiterplatte
Stv. – Steckverbinder

3.3. Grundkonfigurationen des AC A 7100

Von den oben genannten Subsystemen mit lokalem Nebenbus sind auf Grund der physischen Realisierung des Bussystems im Rechnergrundgerät K 1710 des Arbeitsplatzcomputers A 7100 nur 2 Subsysteme gleichzeitig einsetzbar.
Als Finalprodukt im Rahmen des MMS 16 ergibt sich damit bei Einsatz des Zweiportspeichers ZPS K 2071 und des E/A-Subsystems mit KES K 5170 und AFS K 5171/AFP K 5172 eine Konfiguration für einen alphanumerischen Arbeitsplatz mit ABS K 7071 als Anschlußsteuerung für die Bildschirmeinheit. Bei gleichzeitigem Einsatz des obengenannten E/A-Subsystems und des Grafik-Subsystems mit KGS K 7070 und ABG K 7072 ist eine Konfiguration für einen Arbeitsplatz mit grafischer

Tafel 7: Grundkonfigurationen des AC A 7100

Steckplatz-Nr.	alphanumerischer Arbeitsplatz		grafischer Arbeitsplatz	
	Minimal-Konfiguration	Zusatz	Minimal-Konfiguration	Zusatz
7		ZPS od. OPS	ABG	
6	ZVE		KGS	
5	KES		KES	
4	ABS			OPS od. ABS
3		OPS	ZVE	
2	OPS		OPS	
1		ASP od. OPS		ASP od. OPS
10	AFS		AFS	
9		AFP 1		AFP 1

Bildschirmanzeige realisierbar. Tafel 7 gibt einen Überblick über die möglichen Ausrüstungsvarianten für beide Grundkonfigurationen. Auf diesen Grundkonfigurationen des AC A 7100 lassen sich mit Hilfe der zugehörigen Software u.a. folgende Anwendungskomplexe realisieren:

– Zentralisierter Büroarbeitsplatz
– Arbeitsplatz für Konstrukteure und Technologen
– Ingenieurtechnischer Arbeitsplatz
– Programmentwicklungsplatz für MMS 16
– Terminal in Kommunikationssystemen u.v.a.

4. Funktionsprinzipien des AC A 7100

Die Funktionsprinzipien des Arbeitsplatzcomputers A 7100 werden durch die Busprinzipien des standardisierten Systembus sowie die besonderen Funktionsprinzipien der systembestimmenden Schaltkreise aus dem verwendeten Mikroprozessor-Schaltkreissystem K 1810 bestimmt.

4.1. Bauelementebasis des AC A 7100

Bauelementebasis des Arbeitsplatzcomputers A 7100 ist das sowjetische Mikroprozessor-Schaltkreissystem K 1810, das ein LSI-Bauelementespektrum mit dem 16-Bit-Mikroprozessor (CPU) K 1810 WM 86 als Kernstück sowie eine Reihe systembestimmender und peripherer Schaltkreise für den Aufbau von leistungsfähigen 16-Bit-Mikrorechnern mit einer modernen Systemarchitektur und flexibler Anpassung an die verschiedensten Einsatzforderungen umfaßt. Die Schaltkreise des Systems K 1810 besitzen je nach Funktionsumfang verschiedene DIL-Gehäuse mit unterschiedlichen Pinzahlen von 18–40, benötigen nur eine Betriebsspannung von +5 V und sind in unterschiedlichen Halbleitertechnologien hergestellt. Alle Schaltkreise sind vollständig TTL-kompatibel und berücksichtigen die funktionelle Kompatibilität zu den Vorgängersystemen. Aus dem vorhandenen LSI-Bauelementespektrum werden im AC A 7100 die in Tafel 8 aufgeführten Schaltkreise eingesetzt. Darüber hinaus gelangt das Standardbauelementesortiment der Low-Power-Schottky- und Schottky-TTL-Reihe zur Anwendung.

4.2. Busübertragungszyklen

Busübertragungszyklen wie Eingabe, Ausgabe, Speicher Lesen und Speicher Schreiben laufen asynchron nach dem Shakehand-Prinzip ab, wobei ein Signalpaar „Kommando-Quittung" den zeitlichen Ablauf definiert.
Datenübertragungszyklen, an denen jeweils ein Master und ein Slave beteiligt sind, beginnen mit der Aktivierung der Adreßsignale /ADR(13:0)H (sowie der Datensignale /DAT(F:0)H bei Schreib- und Ausgabezyklen) und anschließender Lesekommando- (/MRDC, /IORC) bzw. Schreibkommandoerteilung (/MWTC, /IOWC). Der durch die Adresse angesprochene Slave reagiert je nach Kommando durch die Übernahme der Schreib- oder Ausgabedaten bzw. durch Bereitstellung der Lese- oder Eingabedaten und erteilt das Quittungssignal (/XACK) zur Beendigung des entsprechenden Busübertragungszyklus. Da der AC A 7100 über getrennte Speicher- und E/A-Adreßräume verfügt, existieren für jeden Adreßraum separate Lese- und Schreibkommandosignale (Abb. 5 und 6).
Obwohl der Systembus über einen Adreßsignalumfang von 24 Adreßleitungen (/ADR(17:0)H verfügt, bedienen die Master im AC A 7100 bei Speicherzyklen nur die 20 Adreßsignale /ADR(13:0)H und bei E/A-Zyklen die 16 Adreßsignale /ADR(F:0)H, womit sich der 1 MByte Speicher- und der 64 KByte E/A-Adreßraum ergibt. Ein spezielles Bussignal (BHEN) erlaubt die Steuerung von Byte- und Wortübertragungen. Byteübertragungen laufen unabhängig davon, ob das höherwertige Byte oder das niederwertige Byte übertragen werden soll, stets auf den niederwertigen Datenleitungen /DAT(7:0) ab. Dadurch ist es möglich, auch 8-Bit-Prozessoren oder Speicherbaugruppen mit 8-Bit-Zugriffsbreite an den Systembus anzuschließen bei voller Nutzung des Adreßraumes.
16-Bit-Slaves sind deshalb mit einer Bytevertauschungsschaltung ausgerüstet.

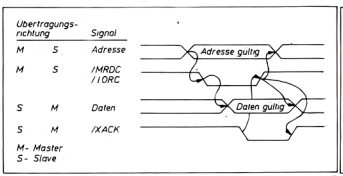
4.3. Speicherorganisation

4.3.1. Speicherorganisation logisch

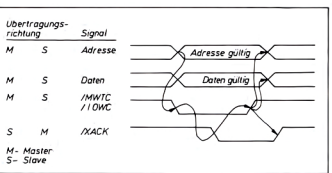
Der Speicher des AC A 7100 ist als lineares Array von 1 Million Bytes mit den Adressen 00000H bis FFFFFH organisiert. Dieses Array kann logisch in Segmente unterteilt werden, die bis zu max. 64 KByte umfassen und an beliebigen 16-Bytegrenzen liegen können. Es werden das
– Codesegment
– Stacksegment
– Datensegment
– Extrasegment

Tafel 8:
K 1810-Bauelementespektrum für AC A 7100

K 1810-Typ	Funktion
K 1810 WM 86	CPU, 16 Bit, 6 MHz
KR 580 WW 51 A	Programmierbares Kommunikationsinterface
KR 580 WI 53	Programmierbarer Intervalltimer
KR 580 WW 55 A	Programmierbares Peripherieinterface
KR 580 WG 57	DMA-Controller
KR 580 WW 59 A	Programmierbarer Interruptkontroller
KR 580 WG 75	Programmierbarer CRT-Controller
KR 580 GF 84	Taktgenerator
KR 580 WG 88	Buscontroller
KR 580 WG 89	Busarbiter



5



6

unterschieden. Jedes Segment wird durch eine Basisadresse charakterisiert, die in einem zugehörigen Segmentregister abgespeichert ist und das niederwertigste Byte innerhalb des jeweiligen Segments adressiert. Ein beliebiges Byte innerhalb eines Segments wird durch Angabe der Segmentbasis und eines Offset adressiert, der die Distanz in Bytes vom Beginn des Segmentes angibt. Somit kann jede Speicherzelle mit einer logischen Adresse angesprochen werden. Die Generierung der physischen Speicheradresse erfolgt durch die CPU (Abb. 7).

Die verschiedenen Segmente können separat, teilweise oder vollständig überlappt oder fortlaufend im Speicher angeordnet werden. Durch diese logische Speicherorganisation ergeben sich effektive Adressierungsmechanismen und Programmorganisationen.

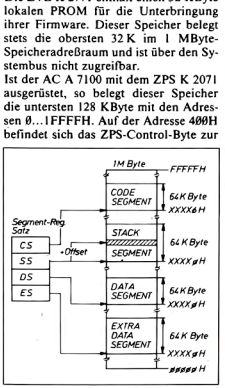
4.3.2. Speicherorganisation physisch

Physisch ist der gesamte Speicheradreßraum des AC A 7100 in zwei Banks von je 512 KByte organisiert. Strukturell ist die eine Bank mit den niederwertigen 8 Datenleitungen verbunden und enthält somit nur Bytes mit geraden Adressen, die andere Bank ist mit den höherwertigen 8 Datenleitungen verbunden und enthält nur Bytes mit ungeraden Adressen. Aus der Sicht der Programmierung kann ein Byte oder ein Wort auf einer beliebigen Adresse abgespeichert werden, d. h. es sind keine Wortgrenzen festgelegt. Ein Byte innerhalb der Bank wird durch die Adressen /ADR(13:1)H ausgewählt. Die Auswahl der entsprechenden Bank erfolgt über die Signale /ADR(0) und /BHE (Abb. 8), das für die Blockierung der höherwertigen Bank verwendet wird. Die richtige Abspeicherung von Bytes in den ZVE-Registern realisiert die CPU K 1810 WM 86 automatisch. Speicherwortzugriffe auf ungerade Adressen werden von der ZVE automatisch in zwei aufeinanderfolgende Bytezugriffe umgewandelt.

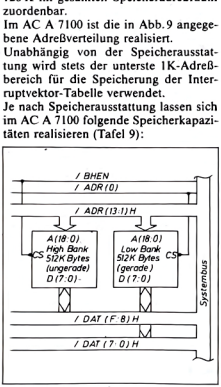
4.3.3. Speicheradreßverteilung

Die ZVE K 2771 enthält einen 32 KByte lokalen PROM für die Unterbringung ihrer Firmware. Dieser Speicher belegt stets die obersten 32 K im 1 MByte-Speicheradreßraum und ist über den Systembus nicht zugreifbar.

Ist der AC A 7100 mit dem ZPS K 2071 ausgerüstet, so belegt dieser Speicher die untersten 128 KByte mit den Adressen Ø...1FFFFH. Auf der Adresse 4ØØH befindet sich das ZPS-Control-Byte zur



7



8

Abb. 5: Lesezyklus

Abb. 6: Schreibzyklus

Abb. 7: Speicherorganisation logisch

Abb. 8: Speicherorganisation physisch

Steuerung der Paritäts- und Interruptschaltung.

Wird der OPS K 3571 eingesetzt, so sind die Systembusadressen in Stufen zu 128 K im gesamten Speicheradreßraum zuordenbar.

Im AC A 7100 ist die in Abb. 9 angegebene Adreßverteilung realisiert.

Unabhängig von der Speicherausstattung wird stets der unterste 1K-Adreßbereich für die Speicherung der Interruptvektor-Tabelle verwendet.

Je nach Speicherausstattung lassen sich im AC A 7100 folgende Speicherkapazitäten realisieren (Tafel 9):

absolute Adresse	Speicher-drauf-teilung	absolute Adresse	
FFFFFH F8000H	32 K Byte lokaler PROM	FFFFFH F8000H	
			F7FFFH
BFFFFH	3. OPS K 3571 256 K		
	Byte	2. OPS K 3571 256 K Byte	9FFFFH
80000H			
7FFFFH	2. OPS K 3571 256 K		60000H
	Byte K 3571	1. OPS	5FFFFH
40000H		256 K	
3FFFFH	1. OPS K 3571 256 K Byte	Byte	20000H
		ZPS K 2071 128 K Byte Control-Byte	1FFFFH
00400H			00400H
003FFH	Interruptvektor-Tabelle (256 Interrupt-vektoren		003FFH
00000H			00000H

Abb. 9: Speicheradreßverteilung im AC A 7100

4.4. Ein-/Ausgabeorganisation

Im AC A 7100 existiert neben dem 1-MByte-Speicheradreßraum ein separater E/A-Adreßraum von 64 KByte. E/A-Adressen werden von der ZVE K 2771 auf den Adreßleitungen /ADR(F:0)H ausgegeben, wobei die Adreßleitungen /ADR(13:10)H inaktiv geschaltet werden. E/A-Einheiten werden auf die gleiche Weise wie Speicherzellen adressiert, d. h. E/A-Zugriffe können als Byte- oder Wortzugriff programmiert werden, jedoch arbeiten alle E/A-Einheiten mit byteweiser Datenübertragung auf den niederwertigen 8 Datenleitungen.

Alle E/A-Einheiten im AC A 7100 belegen im E/A-Adreßraum jeweils einen bestimmten zusammenhängenden Bereich, der bei einigen Moduln durch Einstellung frei wählbar ist.

E/A-Zugriffe zwecks Byteübertragung auf dem Systembus erfolgen sowohl bei gerader als auch ungerader E/A-Adresse immer auf den niederwertigen Datenleitungen. Über eine Swap-Schaltung realisiert die ZVE die notwendige Bytevertauschung. E/A-Wortzugriffe mit gerader Adresse sind erlaubt, jedoch gelangt das höherwertige Byte bei Ausgabe ins Leere.

Bei Eingaben über den Systembus ist in diesem Fall das höherwertige Byte gleich ØØH, bei Eingaben über den lokalen Bus der ZVE gleich FFH. Ungerade E/A-Adressen bei lokalen E/A-Zugriffen sind unerlaubt, da in diesem Fall die CPU K 1810 WM 86 das Byte auf den 8 höherwertigen Datenleitungen erwartet.

Einen Überblick über die E/A-Adreßverteilung im AC A 7100 gibt Tafel 10.

Tafel 9: Speicherausstattung AC A 7100

Kapazität Byte	ZPS K 2071	OPS K 3571
128 K	1 x	–
256 K	–	1 x
384 K	1 x	1 x
512 K	–	2 x
640 K	1 x	2 x
768 K	–	3 x

4.4.1. Ein/Ausgabe-Subsystem

Der Kontroller für Externspeicher (KES K 5170) bildet zusammen mit den Moduln AFS K 5171 sowie AFP K 5172 das intelligente E/A-Subsystem, das als Master am Systembus arbeitet und in seiner Konfiguration auf die Steuerung von Folien- sowie Festplattenspeichern ausgerüstet ist.

Der KES enthält die 8-Bit-CPU UA 880, die einen Subsystem-Bus (UA 880-Nebenbus, Steckverbinder X2) erzeugt, an den AFS und AFP als gerätespezifische Anschlußsteuereinheiten als Slave angeschlossen werden. Die Kopplung des KES an den Systembus erfolgt über einen Window-Bereich mittels einer Swapeinrichtung und der Realisierung der Datenübertragungen nach dem DMA-Verfahren.

Die Kommunikation zwischen System-ZVE und KES erfolgt über Steuer-, Parameter- und Datenblöcke, die von der ZVE im Systemspeicher vorbereitet werden. Die Kommunikationsregeln werden durch eine Softwareschnittstelle definiert, die den Anforderungen der Betriebssysteme SCP 1700, BOS 1810 und des A-7100-Monitorprogramms angepaßt ist.

Nach Erteilung eines Startkommandos durch die System-ZVE bearbeitet der KES die entsprechenden Steuerblöcke, legt das Resultat der Operation im Systemspeicher ab und meldet sich über Interrupt. Die System-ZVE kann dadurch zeitunabhängiger arbeiten und ist von den spezifischen Kontrollerfunktionen entlastet. Diese Funktionen werden durch die im E/A-Subsystem implementierte Firmware realisiert. Die erreichbare E/A-Datenübertragungsrate beträgt 500 KByte/s bei einer OPS-Zugriffszeit von 600 ns. Das E/A-Subsystem erlaubt den Betrieb von jeweils einem von insgesamt vier anschließbaren unterschiedlichen Laufwerkstypen.

4.5. Interruptsystem des AC A 7100

Das Interruptsystem des AC 7100 basiert auf dem Interruptverhalten des CPU-Schaltkreises K 1810 WM 86 auf der ZVE K 2771 in Zusammenarbeit mit dem programmierbaren Interruptcontroller (PIC) KR 580 WN 59 A.

Die Interruptbehandlung ist vektororganisiert auf der Grundlage einer im Speicher vorhandenen Interruptvektor-Tabelle (Tafel 11), die 256 Interruptvektoren in dem unteren 1-K-Bereich enthält.

Die ZVE K 2771 enthält einen Master-PIC und gestattet durch externe Erwei-

Tafel 10:
E/A-Adreßverteilung AC A7100

E/A-Adressen	Einheit	Bemerkung
0000–0001	1. OPS K 3571	Fehlerregister
0002–0003	2. OPS K 3571	Fehlerregister
0040–0041	3. OPS K 3571	Fehlerregister
0042–0043	4. OPS K 3571	Fehlerregister
0080	1. KES	Kanal 1
0081	1. KES	Kanal 2
0082–0083	2 KES (reserv.)	
00C0–00C3	Interruptcontroller K 580 WN 59 A	lokale Ein-/Ausgabe-Ressourcen der ZVE K 2771
00C8–00CF	Programmierbares Peripherie-Interface K 580 WW 55 A	
E/A-Adressen	Einheit	Bemerkung
00D0-00D7	Programmierbarer Intervalltimer K 580 WI 53	
00D8–00DB	Serielles Interface KR 580 WW 51A	
0100–01FF	1. ABS oder KGD	Daten- und Statusregister
0200–02FF	2. ABS oder KGS (reserv.)	
0300–031F	1. ASP	PIO, CTC, SIO
0320–033F	2. ASP (reserv.)	

Tafel 11:
Interruptvektor-Tabelle AC A7100 (Monitor-Initialisierung)

Interruptcode	absolute Adressen Hex.	Interrupt-vektor (IV)	Bemerkung
40–256	00A0–03FF	frei	für Anwender
32–39	0080–009F	PIC-IR 7 : PIC-IR 0	Master-PIC auf ZVE K 2771
5–31	0014–007F	reserviert	für Erweiterungen
4	0010–0013	INTO	Überlauf
3	000C–000F	INT3	1-Byte-Interruptbefehl
2	0008–000B	NMI	nichtmaskierbarer Interrupt
1	0004–0007	SS	Einzelschritt
0	0000–0003	DBZ	Division durch 0

terung mittels Kaskadierung weiterer Slave-PIC über den Systembus, die Anzahl der Interruptebenen auf maximal 64 zu erhöhen. Jede Interruptleitung des Master-PIC (8 Leitungen) kann individuell zur Behandlung eines busvektorisierten (BV) oder eines nicht busvektorisierten Interrupts (NBV) programmiert werden. Um den unterschiedlichen Einsatzfällen Rechnung zu tragen, ist auf der ZVE K 2771 eine Interrupt-Matrix realisiert, die es gestattet, verschiedene Interruptquellen durch Einstellung auszuwählen und dem Master-PIC auf verschiedenen Interruptebenen zur Behandlung anzubieten. Alle Interrupts über den Master-PIC führen in der ZVE zu einem maskierbaren Interrupt (INTR). Darüber hinaus bietet die ZVE die Möglichkeit, Interruptursachen (feste und wählbare) als nichtmaskierbaren Interrupt (NMI) zu behandeln, der jedoch erst durch programmierbare Einstellung freizugeben ist.

Das Interruptsystem des AC A 7100 erlaubt die Behandlung folgender Interrupts:

- Definierte CPU-interne Interrupts
- Programmierbare ZVE-interne Interrupts
- Maskierbare externe Interrupts
- Nichtmaskierbare externe Interrupts.

Das Interruptsystem behandelt fünf definierte CPU-interne Interrupts (siehe Tafel 11), die bei Eintreten oder entsprechender Interruptbedingungen über die festdefinierten Interruptvektoradressen die zugehörige Interruptbehandlungsroutine ohne Ablauf externer Interruptzyklen aufrufen.

Programmierbare ZVE-interne Interrupts werden durch den Nutzer mit Hilfe der 2-Byte-Interruptbefehle INT nn erzeugt, wobei durch nn indirekt die Interruptvektoradresse angegeben wird. Alle maskierbaren externen Interrupts werden von der Systemhardware initiiert, wirken auf den INTR-Eingang der CPU und können über das PIC-Maskenregister sowie das IF-Flag des ZVE Statusregisters maskiert werden.

Nichtmaskierbare externe Interrupts wirken auf den NMI-Eingang der CPU, haben die höchste Priorität und sind über das IF-Flag nicht maskierbar.

4.5.1. Interruptzyklus

Interruptsignale werden über die Systembus-Interruptleitungen /INT(7:0) zur Interrupt-Matrix des ZVE K 2771 übertragen oder von ZVE-internen Interruptquellen geliefert. Die Interrupt-Matrix erlaubt die Auswahl verschiedener Interruptsignale, die auf die Interruptrequest-Eingänge IR (7:0) des Master-PIC gelegt werden (Abb. 10).

Entsprechend einer Initialisierung nimmt er die Prioritätsbewertung vor und meldet über die INTR-Leitung eine akzeptierte Unterbrechungsaufforderung. Die weitere Annahme der Unterbrechungsaufforderung wird von der CPU K 1810 WM 86 gesteuert, die zwei aufeinanderfolgende INTA-Buszyklen initiiert.

– Mit dem ersten INTA-Zyklus wird der Prioritätszustand im Master-PIC eingefroren, d. h. neu entstehende Anforderungen werden nicht mehr beachtet, und es wird die prioritätshöchste Anforderung ermittelt.

– Im zweiten INTA-Zyklus wird ein Interruptcode (abhängig von PIC-Initialisierung) über die niederwertigen Datenleitungen zur CPU übertragen.

Die INTA-Zyklen sind zeitlich durch den CPU-LOCK-Mechanismus untrennbar verbunden.

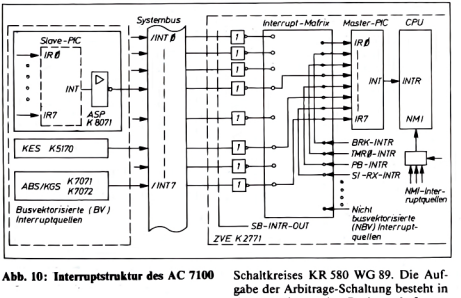


Abb. 10: Interruptstruktur des AC 7100

In Abhängigkeit davon, ob der Interruptcode vom Mater-PIC ZVE-intern zur CPU übertragen wird oder von einem Slave-PIC über den Systembus zur CPU gelangt, unterscheidet man die o.g. BV-und NBV-Interrupts.
Im Fall des BV-Interrupts ist der Slave-PIC über die Adreßleitungen /ADR(A:8) mit den /CAS(2:0)-Signalen des Master-PIC verbunden und sein INTR-Signal gelangt über eine der Systembus-Interruptleitungen zur Interruptmatrix und von dort auf einen Interruptrequest-Eingang IR(7:0) des Master-PIC.
Der übertragene Interruptcode identifiziert eindeutig die Quelle der Interruptanforderung. Die CPU multipliziert den Interruptcode mit vier, wodurch die zugeordnete Interreptvektoradresse gebildet wird, die als Zeiger für den Einsprung in die Interruptvektor-Tabelle dient. Der Interruptvektor enthält eine neue Codesegment-Adresse und einen neuen Befehlszählerstand für den Aufruf der entsprechenden Interruptbehandlungsroutine. Sie muß vor Verwendung natürlich geladen werden (über Monitor oder Betriebssystem).
Die weiteren Abläufe beim Aufruf der Interruptbehandlungsroutine unterscheiden sich nicht von den bei vektororganisierten Interruptsystemen bekannten Vorgängen.

4.6. Busarbitrage

Jeder Busmaster im AC A 7100 (ZVE K 2771, KES K 5170) besitzt eine Arbitrage-Schaltung auf Basis des Arbiter-Schaltkreises KR 580 WG 89. Die Aufgabe der Arbitrage-Schaltung besteht in der Zuweisung der Busherrschaft entsprechend vorgegebener Priorität. Die Arbiter im AC A 7100 sind fertigungsmäßig so verschaltet, daß sie eine serielle Prioritätsentscheidung entsprechend Abb. 11 realisieren und bei einer Bustaktfrequenz von 9,832 MHz die Serienschaltung von maximal 3 Arbitern ermöglichen.
Es wird vorausgesetzt, daß die Systembusbenutzung an jeder beliebigen Programmstelle möglich ist.
Verlangt ein Busmaster die Busherrschaft, während er nicht im Besitz derselben ist, so bewirbt er sich um die Buszuweisung durch Aussenden der Arbitrage-Signale. Die Zuweisung des Busses an einen neuen Master kann frühestens nach Beendigung des auf dem Systembus laufenden Datenübertragungszyklus erfolgen, wobei der prioritätshöchste sich bewerbende Master die Busherrschaft erhält.
Die Zusammenschaltung der Arbiter erfolgt durch eine Daisy-Chain-Kette der Signale /BPRN und /BPRO, indem der /BPRO-Ausgang des höherpriorisierten Arbiters mit dem /BPRN-Eingang des Arbiters mit der nächstniederen Priorität verbunden wird. Das /BPRN-Signal des Arbiters mit der höchsten Priorität bleibt ständig aktiv. Die Prioritätsfolge der Daisy-Chain-Kette /BPRO-/BPRN wird im AC A 7100 durch die Rückverdrahtung vorgegeben (siehe Tafel 6).
Mit dem Systembussignal BUSY zeigt ein Master den Besitz der Busherrschaft an. Die Abgabe der Busherrschaft (nach jedem Buszyklus, nach Anforderung) kann auf jedem Master durch Einstellung festgelegt werden. Mit dem CBRQ-Signal kann auch ein Master mit niederer Priorität ein Gesuch auf die Busherrschaft an den Master mit Busbesitz stellen.

4.7. Arbeit mit dem Zweiportspeicher ZPS K 2071

Bei Einsatz des ZPS K 2071 im AC A 7100 wird dieser über den lokalen ZPS-Nebenbus (Steckverbinder X2) an die ZVE K 2771 und über den Steckverbinder X1 an den Systembus angeschlossen. Der ZPS erlaubt Zugriffe der ZVE über den lokalen Bus sowie Zugriffe eines anderen Busmasters über den Systembus, wobei im letzten Fall die ZVE die Vermittlung des Verkehrs zwischen ZPS und Systembus übernimmt. Damit ist der ZPS eine optionale Erweiterung der ZVE und somit nur gemeinsam mit dieser arbeitsfähig. Der ZPS erlaubt jedoch die Lokalmode-Arbeit der ZVE, d. h. Arbeit mit dem Speicher ohne Systembusbeteiligung. Diese Arbeitsweise ist Voraussetzung für den Aufbau von Multiprozessorsystemen und erlaubt die Arbeit jedes Prozessors mit seinen lokalen Ressourcen.
Zugriffe der ZVE über den lokalen Bus haben eine höhere Priorität als Zugriffe anderer Master über den Systembus, wobei jedoch unter Umständen auf die Beendigung eines laufenden ZPS-Zugriffes eines anderen Busmasters gewartet werden muß. Bei ZVE-Zugriffen liegt der Adreßraum des ZPS im unteren 128 KByte-Bereich des Speicheradreßraumes. Ist der ZPS nicht angeschlossen, erfolgen in diesem Adreßbereich seitens der ZVE automatisch Systembus-Zugriffe. Das Vorhandensein oder Nichtvorhandensein des ZPS in einer Systemkonfiguration erkennt die ZVE K 2771 selbständig. Bei Einsatz des ZPS müssen natürlich ZVE und ZPS auf Steckplätzen untergebracht werden, die einen physisch getrennten Subsystembus bilden.
Fehlerzustände des ZPS werden der ZVE durch Interrupt übermittelt, wobei zur Auswertung zusätzliche Statussignale geliefert werden. Über ein spezielles ZPS-Control-Byte kann das Paritätsverhalten des ZPS gesteuert werden.
Der Einsatz des ZPS K 2071 im AC

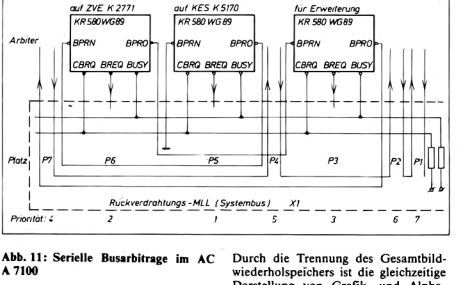


Abb. 11: Serielle Busarbitrage im AC A 7100

A 7100 ist nicht zwingend notwendig, so daß er zwecks Erhöhung der Hauptspeicherkapazität durch einen weiteren OPS K 3571 ersetzt werden kann.

4.8. Grafik-Subsystem

Die Anschlußsteuerung für den Bildschirm (grafisch) ABK K 7072 bildet zusammen mit dem Grafikkontroller KGS 7070 (siehe Abb. 4) das Grafik-Subsystem des AC A 7100.
Der KGS arbeitet als intelligenter Slave am Systembus (X1) und ist über einen Nebenbus (X2) mit der ABG verbunden, wodurch die System-ZVE und der Systembus stark entlastet werden.
Die logische Realisierung des Grafik-Subsystems gewährleistet die funktionelle Kompatibilität zum Modul ABS K 7071 bei alphanumerischer Arbeit. Vor Nutzung des Grafik-Subsystems ist eine spezielle, leistungsfähige Grafikfirmware unter Steuerung der System-ZVE in den KGS zu laden. Die Hardwaremittel des Subsystems, wie eigener Steuerprozessor (UA 880), 64 KByte Programm- und Datenspeicher, getrennte Bildwiederholspeicher (je 64 KByte) für alphanumerische und grafische Darstellung, Interface S2 (V24) für den Anschluß des Grafik-Menütabletts K 6405 sowie IFSS für den Anschluß eines Hardcopy-Druckers, gewährleisten eine leistungsfähige Grafikverarbeitung des AC A 7100, die durch eine Grafik-Grundsoftware (GSX-kompatibel) unterstützt wird (unter Steuerung des Betriebssystems SCP 1700).

Durch die Trennung des Gesamtbildwiederholspeichers ist die gleichzeitige Darstellung von Grafik- und Alphanumerikteilbildern (split-screen) möglich. Die Softwareschnittstelle des KGS K 7070 ist so gestaltet, daß sie den Forderungen des GKS-Standards (Grafik-Kernsystem) entspricht.
Tafel 12 gibt einen allgemeinen Überblick über die Charakteristika des Grafik-Subsystems.

5. Systemanlauf

Nach Spannungszuschaltung erfolgt im AC A 7100 ein automatischer Systemanlauf, der keine Bedienhandlungen erforderlich macht.
Bevor ein Betriebssystem oder ein Nutzerprogramm geladen wird, erfolgt die Abarbeitung des A-7100-Confidence-Test (ACT, Bestandteil des ZVE-Firmware), der die Systeminitialisierung und die Testung aller wesentlichen Systemkomponenten selbst vornimmt oder veranlaßt. Ist die Bildschirmausgabe intakt, so werden dem Bediener die Ergebnisse des Tests auf dem Bildschirm übermittelt.
Im Rahmen dieses Tests erfolgen im einzelnen:

- Ermittlung des angeschlossenen Tastaturtyps, Start des Tastatur-Selbsttests und Abfrage des Ergebnisses
- Kommunikationsaufnahme mit ABS K 7071 oder mit Grafik-Subsystem (KGS K 7070/ABG K 7072), Abfrage der Ergebnisse von deren Selbsttest
- Initialisierung und Test der programmierbaren Schaltkreise auf der ZVE K 2771
- PROM-Prüfsummentest (lokaler PROM auf ZVE K 2771)
- Speicherinitialisierung
  Ermittlung der Speichergröße und Überprüfung der Kommunikationsfähigkeit mit ZPS K 2071 und OPS K 3571
- Prüfung der Speicher-Paritätslogik
- Initialisierung des KES K 5170, Start des KES-Selbsttest und Abfrage des Ergebnisses, Überprüfung der DMA-Fähigkeit
- Test der angeschlossenen Externspeicher.

Werden keine wesentlichen Fehler festgestellt, so wird anschließend versucht, ein Betriebssystem oder ein direkt ladbares Nutzerprogramm, die sich auf einer in einem beliebigen Laufwerk eingelegten Diskette befinden müssen, zu laden. Dazu werden alle Laufwerke in einer bestimmten Reihenfolge abgesucht. Nach dem Laden wird die Steuerung an das geladene Programm übergeben. Wird kein ladbares Programm gefunden oder erlauben wesentliche Fehler kein Laden, so wird die Steuerung an das Monitorprogramm übergeben, das sich mit einer Ausschrift auf dem Bildschirm meldet. Durch Bedienereingriffe (z. B. Betätigen der BREAK-Taste) kann das automatische Laden verhindert und der A-7100-Confidence-Test verkürzt oder modifiziert werden.
Unter Steuerung der Betriebssysteme BOS 1810 und SCP 1700 sind verschiedene formatierte Disketten (8"; 5,25"; FH, MFH; verschiedene Sektor- und Byte/Sektor-Zahlen) ladbar.

6. Ferneinschalten des AC A 7100

Der Arbeitsplatzcomputer A 7100 bietet die Funktion der Netzteilferneinschaltung/-abschaltung über DFÜ, wodurch der Einsatz in unbesetzten Datenstationen möglich wird.
Die Bereitschaft zur Ferneinschaltung des AC A 7100 kann durch einen Befehl ausgelöst werden, der die Betriebsspannungen +5 V, +12 V, −12 V sowie die Lüfter abschaltet.
Der Bereitschaftszustand wird durch eine Zustandsanzeige mit grünem Dauerlicht an der Frontbaugruppe des Rechners angezeigt. Der Vorgang des Ferneinschaltens selbst wird über das Signal Nr. 125 des S2 (V24)-Interfaces

Tafel 12:
Allgemeine Charakteristika des Grafik-Subsystems

- Arbeitsmodi
  Grafikmode mit split-screen Textmode
- Bildwiederholspeicher
  – 2×32 KByte für Grafik
  2×32 KByte für Alphanumerik
- Graustufen 4
- Anzahl der Bildfenster
  – 2 (Grafik-, Textfenster)
  Größe programmierbar
- Textfenster
  Bildfeld
  – 25 Zeilen je 80 Zeichen
  Zeichenraster
  – 7×9 bei Zeichenfeldgröße von 8×16
  Zeichensatz
  – 256, davon 128 ladbar
  lateinischer Zeichensatz gemäß KOI7 HO (ASCII)
  Attribute
  – Blinken, Invertieren, Unterstreichen, weiches Rollen vertikal, erhöhte Helligkeit, Zeichensatzumschaltung
  15 Attributwechsel je Zeile möglich
- Grafikfenster
  Bildfeld
  – 640×400 Bildpunkte
  Schreibgeschwindigkeit
  – ca. 12000 Bildpunkte/s
- Grafische Ausgabeprimitive (lt. GKS)
  – Linien (7 Stärken, 5 Typen), Marker (28 Typen), Kreisbögen (entsprechend Linienattributen), gefüllte Polygonzüge (fill area mit Füllungsarten: solid, hatch, pattern), Text (Qualität STRING)
- Eingabeklassen
  Locator, Pick, Choice, String, Stroke

des Moduls ASP K 8071 ausgelöst und führt zum Einschalten der Stromversorgungsmodule und der Lüfter sowie zum automatischen Anlauf des Rechners.

7.
Netzausfallerkennung

Die Stromversorgungsmoduln des AC A 7100 liefern bei ungenügender oder abgeschalteter Primärspannung ein Power-Fail-Signal, halten ihre Gleichspannungen aber noch mindestens 10 ms. Vor Ablauf dieser Zeit wird über ein spezielles Signal der Rechenvorgang in der ZVE unterbrochen. Alle Moduln werden in den Grundzustand gebracht, wodurch unkontrollierbare Buszyklen zu peripheren Einheiten unterbunden werden.
Nach Spannungswiederkehr wird ein RESET- und das /INIT-Bussignal für mindestens 5 ms erzeugt. Nach Ablauf dieser Zeit beginnt die Zentrale Verarbeitungseinheit mit der Abarbeitung der Firmware ab der durch das RESET-Signal eingestellten Startadresse FFF0H (PROM-Bereich).

8.
Lüfterausfallerkennung

Das Rechnergrundgerät ist zur Realisierung einer Zwangsbelüftung mit zwei Axiallüftern ausgestattet, deren Drehzahl über Miniaturreflexkoppler optisch überwacht wird. Bei Unterschreitung einer bestimmten Drehzahl spricht die Überwachung an und führt über die Signale RESET und/INIT zum Abschalten der gesamten Logik des AC A 7100. Da die Frontbaugruppe über eine separate Stromversorgung verfügt, wird dieser Zustand durch Blinken der gelben Bereitschafts-Anzeige signalisiert.

9.
Schlußbemerkungen

Der vorgestellte neue Arbeitsplatzcomputer A 7100 ist aufgrund seiner modernen Systemarchitektur und durchgängigen modularen Gestaltung ein leistungsstarker, grafikfähiger 16-Bit-Mikrorechner, der dem internationalen Entwicklungsstand in dieser Rechnerklasse entspricht. Seine zukunftsorientierte Konzeption, das verwendete Mikroprozessor-Schaltkreissystem, der standardisierte Systembus und seine konstruktive Gestaltung ermöglichen den Ausbau des Erzeugnisses, die Steigerung seiner Leistungsfähigkeit und durch universelle Konfigurierbarkeit bezüglich der Hard- und Software die Anpassung an die verschiedensten Einsatzfälle.
Die Entwicklung des Arbeitsplatzcomputers A 7100 erfolgte in Abstimmung mit einem sowjetischen Partner, dessen äquivalentes Erzeugnis im Rahmen des SKR die Bezeichnung CM 1810 trägt. Durch diese abgestimmte Entwicklung sowohl der Hardware- als auch der Softwarekomponenten konnte die volle Kompatibilität beider Erzeugnisse auf Anwenderprogrammniveau erreicht werden.
Neben der Nutzung von A-7100-spezifischen Baugruppen ist aufgrund der exakt definierten Interface-Schnittstellen auch die Nutzung von MMS 16-Baugruppen anderer Produzenten sowie der Einsatz von branchenspezifischen Baugruppen und Gerätekomponenten möglich, wodurch der AC A 7100 die Grundlage für weitere Finalerzeugnisse bildet.
So werden z. B. auf der Basis des AC A 7100 das Interaktive Grafische Terminal K 8918 des VEB Kombinat Robotron als weiteres Finalerzeugnis sowie im Kombinat Carl-Zeiß Jena spezielle Steuerungssysteme und Meßgerätekomplexe realisiert.
In Verbindung mit der in der DDR zur Verfügung stehenden CAD-Peripherie sowie der flexiblen Grafiksteuerung des AC A 7100 ist der Einsatz des Rechners für viele Aufgaben der automatisierten, rechnergestützten Konstruktion und des Entwurfs möglich, wodurch mit dem Arbeitsplatzcomputer A 7100 ein wesentlicher Beitrag bei der Durchsetzung der Schlüsseltechnologie CAD/CAM geleistet wird.
Der AC A 7100 erweitert das Anwendungsspektrum von Mikrorechnern im Vergleich mit den vorhandenen 8-Bit-Büro- und Personalcomputern, wobei u. a. folgende Effekte erzielt werden:

- eine ZVE-Leistungssteigerung um den Faktor 5
- eine Erweiterung der Hauptspeicherkapazität auf das 12fache
- eine um 20 Prozent geringere Energieaufnahme
- eine Gewichtsreduzierung um 10 Prozent durch Einsatz einer neuen Basiskonstruktion
- eine Volumenreduzierung um etwa 25 Prozent bei den Stromversorgungsmoduln bei gleicher Leistung und einer Wirkungsgraderhöhung um etwa 5 Prozent.

/1/ Junge, S., Keller, D.:
Das Mikrorechnermodulsystem 16 und sein Einsatz im Arbeitsplatzcomputer robotron A 7100. Neue Techn. Büro (Ausgabe dtsch. Sprache) Berlin 29 (1985) 3, S. 81–87
/2/ Riegel, H.:
Arbeitsplatzcomputer A 7100
Hardware-Systemüberblick in WIB Nr. 25,
VEB Robotron-Elektronik Dresden (1986), S. 3–41

Das Betriebssystem SCP 1700

Ulrich Heckel
VEB Robotron-Elektronik Dresden

1. Allgemeines

SCPS 1700 (Single User Control Program) ist ein externspeicherorientiertes Einzelnutzerbetriebssystem, das speziell für den Einsatz des A 7100 im kommerziellen Bereich als Personalcomputer vorgesehen ist. Typische Anwendungsgebiete mit SCP 1700 als Basisbetriebssystem sind:
– Buchung, Fakturierung, Abrechnung
– wissenschaftlich-technische Berechnungen
– Textverarbeitung
– Programmentwicklung.
Zum Betriebssystem SCP 1520 für die 8-bit-Bürocomputer A 5120/30 und PC 1715 besteht hinsichtlich der unterstützten Diskettenformate und des Dateiaufbaus Kompatibilität.
SCP 1700 ist modular aufgebaut und besteht aus den Teilen (Abb. 1):
– Steuerprogrammlader
– Steuerprogramm (Grundpaket)
– externe Kommandos (Dienstprogramme)
– Programmpaket für modulare Programmierung
– Grafikerweiterung für SCP 1700
– Compiler und Interpreter
– Standardsoftware.

2. Steuerprogrammlader LDSCP

Der Steuerprogrammlader befindet sich auf den ersten Spuren (Systemspuren) einer SCP 1700-Systemdiskette und wird vom Monitorprogramm des A 7100 nach Eingabe des Kommandos B (Boot) in den Speicher geladen und gestartet. Er lädt dann seinerseits das SCP 1700-Steuerprogramm, das als Datei SCP.SYS auf der Diskette enthalten sein muß und übergibt ihm die Steuerung. Alle weiteren Programme arbeiten unter der Regie des Steuerprogramms.
Zum Kopieren des Steuerprogrammladers auf die ersten Spuren einer Systemdiskette wird das Dienstprogramm LDCOPY (LOADER COPY) bereitgestellt.

3. SCP 1700-Steuerprogramm (SCPX 1700)

SCPX ist der hauptspeicherresidente Teil des Betriebssystems SCP 1700. Er besteht aus 3 Moduln:
– CCP (Command Console Processor)
– BDOS (Basic Disk Operating System)
– BIOS (Basis Input/Output System).
Im Unterschied zu SCP 1520 ist auch der Teil CCP ständig hauptspeicherresident, so daß beim Warmstart (CONTROL-C) nichts von der Diskette nachgeladen werden muß. Die Lage von SCPX 1700 im Operationsspeicher ist aus Abb. 2 ersichtlich.

3.1. BIOS

Das BIOS ist der hardwareabhängige Teil des Steuerprogramms, d. h. beim Einfügen oder Ändern von Gerätebedienroutinen im BIOS kann SCP 1700 an eine nutzerspezifische Gerätekonfiguration angepaßt werden.
BIOS verfügt über Eintrittspunkte für 21 Subroutinen. Diese Eintrittspunkte werden vom Nutzerprogramm in der Regel indirekt über BDOS-Rufe angesprochen, können aber mit dem Ruf DIRECT BIOS CALL auch direkt benutzt werden.
Periphere Geräte werden im SCP 1700 als logische Geräte betrachtet. Man unterscheidet
– die zeichenorientierten Geräte CONSOLE (Konsolengerät), LIST (Listengerät), AXI (allg. Eingabegerät), AXO (allg. Ausgabegerät) von den
– dateiorientierten Geräten A... P.
Die Zuordnung zu den physischen Geräten erfolgt im BIOS. Jedem zeichenorientierten Gerät können bis zu vier physische Geräte zugeordnet werden. Die aktuelle Zuordnung wird über das sogenannte IOBYTE gesteuert. Ein dateiorientiertes Gerät A... P bezieht sich im allgemeinen auf ein physisches Gerät (Floppy-Laufwerk). Es ist jedoch auch möglich, ein physisches Gerät in mehrere logische Geräte aufzuteilen. Die Parameter für den Zugriff zu den dateiorientierten Geräten werden innerhalb des BIOS in speziellen Datenstrukturen gespeichert.

Mit den SCP 1700-BIOS werden die in Abb. 3 dargestellten Geräte bzw. Geräteschnittstellen des A 7100 bedient.
Die Speicherdiskette ist ein Bereich im Operationsspeicher des A 7100, der logisch vom Betriebssystem wie ein Diskettenspeicher verwaltet wird. Die Nutzung dieser Speicherdiskette als Arbeitsdiskette erhöht aufgrund der geringen Zugriffszeiten die Arbeitseffektivität am A 7100 beträchtlich.

3.2. BDOS

3.2.1. BDOS-Funktionen

BDOS enthält etwa 50 Betriebssystemfunktionen für
– einfache Zeichen- Ein-/Ausgabe (sie beziehen sich auf die logischen Geräte CONSOLE, LIST, AXI, AXO)
– Dateioperationen (diese Funktionen beziehen sich auf die logischen Geräte A bis P und beinhalten unter anderem die Operationen Eröffnen, Schließen, Löschen und Umbenennen von Dateien, sequentielles und direktes Lesen oder Schreiben von Dateien sowie Operationen mit dem Dateiverzeichnis. Die logische Satzlänge bei einer Lese-/Schreiboperation ist generell 128 Byte).
– Speicherverwaltung
– Programm laden.
Der Eintritt ins BDOS erfolgt über den Softwareinterrupt 224. Der Funktionscode zur Auswahl einer bestimmten BDOS-Funktion und die notwendigen Parameter werden in Registern übergeben:

BDOS-Eintritt	BDOS-Ausgang
CL Funktionscode	AL Rückkehrwerte vom Typ BYTE
DL BYTE-Parameter	AX od. BX Rückkehrwerte vom Typ RORD
DX WORD-Parameter	ES: BX Doppelwortadressen ES Segmentadresse BX Offset
DS Segmentadresse des Datensegments im aufrufenden Programm	

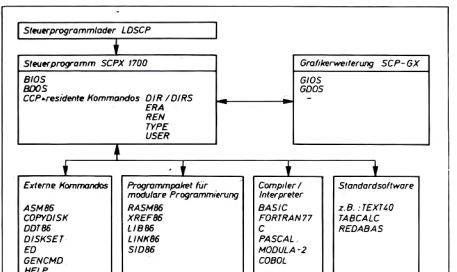
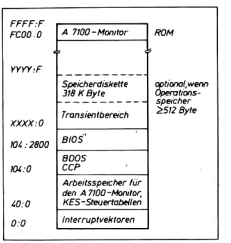


Abb. 1 Bestandteile von SCP 1700

Abb. 2 Aufteilung des A 7100-Operationsspeichers für SCP 1700



3.2.2. Speicherverwaltung

Der in der vorliegenden Hardware-Konfiguration vorhandene und nicht vom Steuerprogramm und der Speicherdiskette belegte RAM-Bereich (Transientbereich) ist in einer Tabelle im BIOS erfaßt. Der RAM-Bereich muß nicht zusammenhängend sein. Die Tabelle kann Eintragungen über maximal acht Teilbereiche enthalten.

Über diesen statisch erfaßten Speicher realisiert SCPX 1700 eine dynamische Speicherverwaltung. Der verfügbare Speicher kann in acht Regionen aufgeteilt werden. Eine Anforderung zum Reservieren einer Speicherregion erfolgt entweder implizit beim Laden eines Programms mit dem CCP oder explizit mit einem Nutzerprogramm mittels entsprechender BDOS-Rufe.

3.2.3. Datenträgerorganisation

Diskettenformate

Auf den am A 7100 anschließbaren Diskettenlaufwerken sind folgende Diskettenarten einzusetzen:

– K 5600.20 5 ¼"-Disketten, einseitig, MFM, 96 tpi und

– K 5602.10 8"-Disketten, einseitig, FM.

Standardmäßig wird beim SCP 1700 mit sogenannten „Hausformaten" gearbeitet, die von den „Hausformaten" des Betriebssystems SCP 1520 abgeleitet wurden:

– K 5600.20 Spur 0 (FM) 16 Sektoren à 128 Byte, Spur 1–79 16 Sektoren à 256 Byte, 306 KByte Anwenderkapazität

– K 5602.10 Spur 0 26 Sektoren à 128 Byte, Spur 1–76 4 Sektoren à 1024 Byte, 294 KByte Anwenderkapazität.

Mit dem Uminitialisieren der BIOS-Steuertabellen für den Externspeicherzugriff können auch abweichende Formate verarbeitet werden (siehe Kommando DISKSET Abschn. 4.3.).

Diskettenverwaltung

Der Foliespeicher wird vom SCPX 1700 dynamisch verwaltet. Beim allerersten Zugriff auf ein Laufwerk durchsucht SCPX 1700 das Verzeichnis der dort vorhandenen Diskette und legt eine Tabelle darüber an, welche Abschnitte der Diskette mit gültigen Aufzeichnungen belegt und welche noch frei sind. Diese Tabelle dient als Grundlage zum Berechnen des freien Speicherplatzes und zur Auswahl neuer Aufzeichnungsabschnitte. Zusätzlich wird eine Prüfsumme über den Inhalt jedes Verzeichnisabschnitts berechnet und in einer besonderen Prüftabelle abgelegt. Anhand dieser Prüftabelle kann SCPX 1700 beim Zugriff auf ein Laufwerk feststellen, ob ein Diskettenwechsel stattgefunden hat.

Dateiverwaltung

Der Zugriff zu den Dateien erfolgt über ein Dateiverzeichnis. Jeder Verzeichniseintrag enthält außer dem Dateinamen Angaben über die von der Datei belegten Speicherblöcke, über die Zugehörigkeit zu einer bestimmten Nutzergruppe sowie über bestimmte Dateiattribute (z. B. Nur-Lese-Dateien).

Die Nutzergruppe wird mit der Zuweisung einer Nutzernummer beim Anlegen einer Datei festgelegt. Alle Systemprogramme, außer PIP, haben nur zu den Dateien Zugriff, die die Nutzernummer besitzen, unter der augenblicklich gearbeitet wird.

logisches Gerät	physisches Gerät
CONSOLE	A 7100-Konsole (Bildschirm, Tastatur) Bildschirmterminal (IFSS, V24)
LIST	Drucker K 6311 (Centronics) Drucker robotrón 1152 bzw. 1157 (IFSS und IFSP) A 7100-Bildschirm
AXI	A 7100-Tastatur Eingabegerät (IFSS, V24)
AXO	A 7100-Bildschirm Ausgabegeräte (IFSS, V24 und IFSP)
A, B	Diskettenlaufwerke K 5600.20 5 ¼", 80 Spuren, einseitig
C, D	Diskettenlaufwerke K 5600.20 5 ¼", 80 Spuren, einseitig oder Diskettenlaufwerke K 5602.10 8", 77 Spuren, einseitig, einfache Dichte
E	Speicherdiskette (ab 512 KByte Operationsspeicher)

Abb.3 Geräte bzw. Geräteschnittstellen des A 7100

3.3. CCP

Der CCP ist das Programm, das den Nutzerdialog führt. Er nimmt Kommandoanforderungen des Nutzers entgegen und bearbeitet sie entweder selbst oder lädt entsprechende Programme von der Diskette. Die Bereitschaft zur Entgegennahme eines Kommandos meldet CCP durch Ausgabe der Zeichenfolge

d>

d steht für den Namen des Bezugslaufwerkes (A, ... P), d. h. desjenigen Laufwerkes, auf den sich alle Dateinamen in der Kommandozeile beziehen, wenn sie keine explizite Laufwerksangabe enthalten.

SCP 1700 unterscheidet zwei Arten von Kommandos:

– Residente Kommandos – sind im CCP selbst enthaltene Funktionen, die bei Bedarf direkt aufgerufen werden können (z. B.: DIR, ERA, REN)

– Transiente Kommandos – sind lauffähige Programme, die als Dateien des Typs CMD auf der Diskette stehen und vor dem Abarbeiten in den Transientbereich geladen werden müssen (Dienstprogramme, Übersetzer, Nutzerprogramme).

DAS CMD-Format ist das vom SCP-Lader geforderte Ladeformat. Es besteht aus einer 128 Byte langen Kopfinformation, gefolgt von bis zu acht Speichersektionen:

CODE DAT EXTRA STACK X1 X2 X3 X4.

Im Kopf werden diese Sektionen mit der Ladeadresse, der Länge sowie mit dem minimalen und maximalen Speicherbedarf beschrieben. Der SCP-Lader benutzt diese Informationen, um

– Speicher für das Programm zu reservieren

– die Segmentregister zu initialisieren und um

– die sogenannte Basis-Seite aufzubauen.

4. SCP 1700-Kommandos

Allgemeiner Aufbau einer Kommandozeile

Eine Kommandozeile besteht aus einem Kommandoschlüsselwort und einem Parameterteil. Das Kommandoschlüsselwort bezeichnet das gewünschte Kommando und ist bei externen Kommandos identisch mit dem Dateinamen der zu ladenden CMD-Datei. Der Parameterteil besteht aus Dateispezifikationen und Optionen entsprechend den Kommandos.

Beispiel:

```
ERA DATEI.BAK
Löschen (ERASE) der Datei DATEI.BAK
```

Gültige Dateispezifikationen im SCP 1700 haben folgenden allgemeinen Aufbau:

```
d:filename.typ
d:filename
    filename.typ
    filename
Beispiel:
A:DATEI1.A86
A:DATEI1
    DATEI1.A86
    DATEI1
```

d – Laufwerksbezeichnung A ... P

filename – Dateiname, bestehend aus 1–8 alphanumerischen Zeichen

typ – Dateityp, bestehend aus 1–3 alphanumerischen Zeichen.

Bestimmte Kommandos erlauben mit der Angabe von Dateigruppensymbolen (wildcards) in filename und typ auch die Auswahl mehrerer Dateien:

? steht für ein einzelnes zulässiges Zeichen an der betreffenden Stelle

* steht für eine Gruppe von Zeichen an der betreffenden Stelle.

In den folgenden Kapiteln werden die unter SCP 1700 verfügbaren Kommandos kurz charakterisiert.

Assembler ASM86 (extern)

Aufgaben

Der Assembler übersetzt ASM86-Quellprogramme in Objektprogramme im Hexadezimalcode, die von GENCMD weiterverarbeitet werden.

Arbeitsweise

Der Assembler arbeitet in drei Durchläufen und erzeugt außer der Hex-Codedatei ein Assemblerprotokoll sowie eine Liste der im Quell-Programm verwendeten Symbole. Mit Optionen kann der Nutzer steuern, welche Geräte für welche Dateien benutzt und ob Ausgabedateien unterdrückt werden sollen.

ASM86 bietet Direktiven zur

– Programmsegmentierung (absolute und verschiebliche Segmente)

– Manipulation des Speicherplatzzählers

– bedingten Assemblierung

– Definition von Symbolen

– Speicherplatzreservierung und -initialisierung

– Einbeziehung von INCLUDE-Dateien

– Listensteuerung.

Außerdem kann der Nutzer mit dem Verwenden von Code-Makros „eigene" Maschinenbefehle definieren.

Diskettenkopierprogramm COPYDISK (extern)

Aufgaben

COPYDISK kopiert alle Informationen einer Diskette auf eine andere einschließlich der SCP 1700-Systemspuren, falls sie auf der Quelldiskette vorhanden sind.

Arbeitsweise

Der Kopiervorgang erfolgt spurweise. Voraussetzung für die Arbeit von COPYDISK ist, daß die Zieldiskette forma-

5 ¼", doppelte Dichte		8", 77 Spuren	
40 Spuren 1)	80 Spuren	einfache Dichte	doppelte Dichte
26×128 Bytes/Spur 2) 16×256 3) 8×512 9×512 4×1024 5×1024	26×128 2) 16×256 3) 8×512 9×512 4×1024 5×1024	26×128 2) 15×256 8×512 4×1024 3)	26×256 15×512 8×1024 4)

Abb. 4 Einstellbare Formate durch DISKET

tiert ist und die gleichen Parameter wie die Quelldiskette aufweist. COPYDISK erlaubt das Kopieren von Disketten, die abweichend von den „Hausformaten" initialisiert sind.

Testhilfe DDT86 (extern)

Aufgaben

DDT86 dient zum Testen von Nutzerprogrammen, die im CMD-Format auf Diskette bereitstehen müssen. Die Nutzung von DDT86 ist in erster Linie zum Testen von in Assemblersprache geschriebenen Programmen sinnvoll.

Arbeitsweise

Nutzerprogramme können entweder zusammen mit DDT86 oder von DDT86-Kommandos geladen werden. Über die Bedienerkommunikation können folgende Leistungen von DDT86 angefordert werden:

– Ausgabe und Ändern von Register- und Speicherinhalten
– Auslisten von Speicherinhalten in mnemonischer Form
– Eingabe von Assembleranweisungen in mnemonischer Form
– Vergleichen von Steuerblöcken
– Verschieben von Speicherblöcken
– Berechnen der Summe und Differenz zweier Hexadezimalzahlen
– Steuern der Abarbeitung des Nutzerprogramms mit und ohne Protokollierung (schrittweises Abarbeiten, Setzen von Unterbrechungspunkten).

Anzeige von Dateiverzeichnissen DIR/DIRS (resident)

Aufgaben

DIR zeigt die Namen aller im Parameterteil beschriebenen Dateien an, die im Dateiverzeichnis des spezifizierten Gerätes enthalten sind und das Attribut DIR haben. DIRS ist ein äquivalentes Kommando für Dateien mit dem Attribut SYS.

Ändern der Diskettenzugriffsparameter DISKSET (extern)

Aufgaben

SCP 1700 verarbeitet standardmäßig die „SCP-Hausformate". Mit DISKSET können einige Diskettenzugriffsparameter temporär geändert werden, um den Zugriff auf abweichend formatierte Disketten zu gewährleisten.

Arbeitsweise

Die änderbaren Parameter werden dem Nutzer über Menü-Angebote zur Auswahl gestellt. Die Änderungen bleiben gültig bis zum erneuten Ändern mit DISKSET oder zum Neustart von SCP 1700.

Die zur Zeit mit DISKSET einstellbaren Formate sind in Abb. 4 dargestellt. Für alle Formate sind außerdem die logischen Parameter diranz (maximale Anzahl von Verzeichniseinträgen) zwischen 64 und 128 sowie sysoff (Anzahl von Systemspuren) von eins bis neun auswählbar.

① Auf K 5600.10 geschriebene Disketten sind auf K 5600.20 lesbar. Der umgekehrte Weg ist hardware-bedingt nicht möglich.

② Skewfaktor = 6, CP/M-Standardformat für diranz = 64 und sysoff = 2

③ „SCP-Hausformat" für diranz = 64 und sysoff = 3

④ „SCP-Hausformat" für diranz = 128 und sysoff = 2

Editor ED

Aufgaben

ED ist ein Programm zum Anlegen und Verändern von Textdateien. Mit seiner Hilfe lassen sich

– Textdateien erzeugen
– Texte in vorhandene Dateien einfügen
– Texte in vorhandenen Dateien löschen
– Texte in vorhandenen Dateien umgruppieren und mit Einschränkungen auch
– Textausschnitte erstellen und
– vorhandene Textdateien zusammenfügen.

Arbeitsweise

Mit ED wird der Text aus der Originaldatei über einen Pufferspeicher in eine Zwischendatei kopiert. Dieser Kopiervorgang kann mit ED-Kommandos interaktiv gesteuert werden. Man kann

– eine gewünschte Anzahl von Zeilen in den Puffer einlesen
– eine gewünschte Anzahl von Zeilen in die Zwischendatei schreiben
– den Text, solange er im Pufferspeicher ist, wie folgt bearbeiten:

- Einen neuen Text über die Tastatur oder von einer Datei aus einfügen
- Textausschnitte in eine Datei schreiben
- beliebige Textstellen aufsuchen
- beliebige Textstellen löschen
- beliebige Textstellen verändern.

Wenn der Nutzer den Editiervorgang beendet hat, überträgt ED den Inhalt des Puffers, gefolgt von dem noch unbearbeiteten Text aus der Quelldatei, in die Zwischendatei. Anschließend werden die Dateien umbenannt. Die Quelldatei, deren Inhalt während des Editiervorgangs nicht verändert wird, bleibt dem Nutzer unter ihrem ursprünglichen Dateinamen (mit der Dateityperweiterung BAK) erhalten. Die Zwischendatei erhält den Dateinamen sowie die Dateityperweiterung der Quelldatei.

Löschen von Dateien ERA (resident)

Aufgaben

Das Kommando ERA dient zum Löschen einer Datei oder Gruppen von Dateien.

Ladeformatgenerierungsprogramm GENCMD (extern)

Aufgaben

GENCMD transformiert ein Objektprogramm aus dem vom Assembler ASM86 erzeugten Hexadezimalcode (H86) in das SCP 1700-Ladeformat (CMD).

Arbeitsweise

Die vom GENCMD erzeugte CMD-Datei besteht aus einem 128 Byte langen Vorsatz, gefolgt vom Speicherbild des Programms. Der Vorsatz beinhaltet alle

notwendigen Informationen über die im Speicherabbild enthaltenen Codesektionen. Gewonnen werden diese Informationen zum einen aus speziellen Sätzen der Hexadezimalcode-Datei und zum anderen aus Optionen beim Aufruf von GENCMD.

Bedienerhilfsprogramm HELP (extern)

Aufgaben

Mit HELP kann sich der Nutzer Informationen über alle möglichen SCP 1700-Kommandos anzeigen lassen.

Arbeitsweise

Das HELP-Kommando arbeitet über einer speziellen Textdatei HELP.HLP. Mit der Eingabe festgelegter Begriffe (topics) und Unterbegriffe (subtropics) in der Kommandozeile zum Aufruf von HELP bzw. auf Anfrage von HELP kann der Nutzer gezielt Informationen über ein bestimmtes Kommando anfordern. Der Nutzer hat die Möglichkeit, die Textdatei mit eigenen Begriffen zu erweitern.

Diskettenformatierprogramm INIT (extern)

Aufgaben

Die Disketten müssen vor dem ersten Einsatz formatiert, d. h. zur Datenaufnahme bereitgemacht werden. Diese Funktion wird von dem Dienstprogramm INIT realisiert. Außerdem initialisiert INIT den Diskettenabschnitt, der von SCP 1700 als Verzeichnisbereich benutzt wird.

Arbeitsweise

Zum Formatisieren von Datenträgern werden vom SCPX 1700 keine Rufe bereitgestellt. Der Datentransport muß daher über direkte Ein-/Ausgabe erfolgen. Defekte Spuren werden beim Initialisieren mit einer Sperrdatei belegt.

Laderkopierprogramm LDCOPY (extern)

Aufgaben

LDCOPY dient dem Erstellen von SCP 1700-Systemdisketten. Mit seiner Hilfe kann der SCP 1700-Steuerprogrammlader LDSCP auf die ersten Spuren (Systemspuren) einer formatierten Diskette übertragen werden.

Arbeitsweise

Beim Aufruf von LDCOPY kann der Name einer CMD-Datei angegeben werden. Der Steuerprogrammlader wird dann von dort in den Hauptspeicher geladen. Ist kein Dateiname angegeben, fordert das Kopierprogramm vom Nutzer die Eingabe einer Laufwerksbezeichnung und lädt die ersten beiden Spuren der dort befindlichen Diskette. Danach erfragt LDCOPY die Laufwerksbezeichnung der Zieldiskette und überträgt den zuvor eingelesenen Speicherinhalt auf die Systemspuren dieser Diskette.

Dateitransferprogramm PIP (extern)

Aufgaben

PIP ist ein Programm zum Informationsaustausch zwischen den Peripherieeinheiten, wobei unter Peripherieeinheiten Folienspeicher und die über die Ein-/Ausgabekanäle CONSOLE, AXI, AXO und LIST angeschlossenen Geräte zu verstehen sind. Mit Hilfe von PIP lassen sich Daten von einer derartigen Peripherieeinheit zu einer anderen übertragen. Beispielsweise ist es möglich

– eine Datei in eine andere zu kopieren
– eine ganze Diskette zu kopieren
– den Inhalt einer Textdatei auszudrucken
– den Inhalt einer Datei auf den Bildschirm auszulisten
– mehrere Einzeldateien zu einer Gesamtdatei zusammenzufügen
– Dateiauszüge zu erstellen und
– Textdateien (Groß- in Kleinschreibung, Verändern der Seitenlänge, Verändern der Zeilenlänge u. ä.) umzuformen.

Arbeitsweise

Die Funktionen von PIP werden mit der Angabe entsprechender Optionen in der Kommandozeile aufgerufen.

Umbenennen von Dateien REN (resident)

Aufgaben

Mit dem Kommando REN kann der Name einer existierenden Datei geändert werden.

Systemstatusprogramm STAT (extern)

Aufgaben

STAT dient dazu, sich einen Überblick über die wichtigsten Systemeigenschaften zu verschaffen und diese, soweit möglich, gezielt zu verändern. Dies betrifft sowohl die Peripheriegeräte (Ein- und Ausgabegeräte, Massenspeichereinheiten) als auch den Speicherbedarf und die Merkmale von Dateien.

Arbeitsweise

Die verschiedenen Dienste werden mit der Angabe von Optionen in der Kommandozeile zum Aktivieren von STAT angefordert. Mit STAT VAL: werden dem Nutzer alle möglichen Optionen von STAT ausgelistet. Folgende Dienste werden geboten:

– Ausgabe einer Übersicht der auf einer Diskette vorliegenden Benutzerbereiche (Nutzernummern)
– Auslisten der aktuellen Zuordnung von E/A-Kanälen und Geräten
– Zuweisen von Geräten zu E/A-Kanälen
– Information über den freien Speicherplatz auf einer Diskette
– Auslisten der Merkmale und des belegten Speicherplatzes von Dateien
– Setzen von Dateimerkmalen
– Auslisten der Eigenschaften der derzeit aktiven Disketten.

Stapelverarbeitungsprogramm SUBMIT (extern)

Aufgaben

Mit SUBMIT hat der Nutzer die Möglichkeit, Kommandodateien zur automatischen Abarbeitung an SCPX 1700 zu übergeben.

Arbeitsweise

Eine Kommmandodatei ist eine Folge von gültigen SCP 1700-Kommandos. Sie muß den Dateityp SUB haben. Die Kommandos können formale Parameter enthalten, die beim Aufruf von SUBMIT durch in der Kommandozeile anzugebende aktuelle Parameter ersetzt werden. Das SUBMIT-Kommando selbst kann in Kommandodateien nur als letztes Kommando stehen.

Anzeigen von Textdateien TYPE (resident)

Aufgaben

Mit dem Kommando TYPE kann der Inhalt einer ASCII-Datei auf dem CONSOLE-Gerät angezeigt werden.

Nutzerbereichsumschaltung USER (resident)

Aufgaben

Mit dem Kommando USER kann der aktuelle Nutzerbereich (0 bis 15) für den Zugriff zu Externspeichern eingestellt werden. Die Aufteilung von Ex-

ternspeichern in Nutzerbereiche ist vor allem dann sinnvoll, wenn mehrere Nutzer ein Speichermedium (z. B. eine Festplatte) gemeinsam nutzen oder wenn auf dem Speichermedium eines Nutzers Dateien für unterschiedliche Aufgabenklassen getrennt verwaltet werden sollen.
Dienstprogramme haben in der Regel nur zu den Dateien Zugriff, die die Nutzernummer besitzen, unter der augenblicklich gearbeitet wird.
Dateien mit dem Attribut SYS im Nutzerbereich 0 sind von allen Nutzerbereichen aus als Nur-Lese-Dateien verfügbar.

SCP 1700-Steuerzeichen

SPC 1700 unterstützt eine Reihe von Steuerzeichen, die als Kombination der Taste CONTROL mit einer Buchstabentaste eingebbar sind. Sie dienen in der Hauptsache zum Editieren bei der Kommandoeingabe. Drei Steuerzeichen haben besondere Bedeutung:
CTRL-C Beenden des aktuellen Programms bzw. bei Eingabe nach Kommandoanforderung des CCP Reinitialisierung des Systems
CTRL-P Parallelschalten des Druckers zur Konsolenausgabe bis zum nächsten CTRL-P
CRTL-S Anhalten einer laufenden Konsolenausgabe bis zum nächsten Tastendruck.

5.
Programmpaket für modulare Programmierung

Das Programmpaket besteht aus den Teilen:
– Assembler RASM86
– Cross-Referenzlistenausgabe XREF86
– Bibliothekar LIB86
– Linker LINK86.
Diese Programme bilden die Grundlage für die Nutzung des A 7100 als Programmentwicklungssystem. Der FORTRAN-77-Compiler für SCP 1700 und weitere geplante Sprachübersetzer bauen darauf auf.

Assembler RASM86

Aufgaben

Der Assembler übersetzt RASM86-Quellprogramme in Objektcode, der von LINK86 weiterverarbeitet wird. Im Gegensatz zu ASM86 besteht die Möglichkeit, mit Hilfe globaler Symbole Verbindungen zu anderen unabhängig übersetzten Programmen herzustellen. Für ASM86 geschriebene Quellen sind durch RASM86 assemblierbar.

Arbeitsweise

RASM86 verarbeitet eine RASM86-Assemblercodedatei in drei Pässen und produziert daraus eine Objektcodedatei. Außerdem können zwei weitere Dateien erzeugt werden – eine Assemblerliste mit Fehlermitteilungen und eine Liste der definierten Symbole. Beim Aufruf von RASM86 können (mit Parametern) die Ausgabedateien unterschiedlichen logischen Geräten zugewiesen oder auch unterdrückt werden.
RASM86 bietet Direktiven zur
– Programmsegmentierung (mit der Angabe von Segmentparametern können dem Linker Informationen über besondere Eigenschaften der jeweiligen Segmente übermittelt werden)
– Speicherplatzreservierung und -initialisierung
– Manipulation des Speicherplatzzählers
– bedingten Assemblierung
– Definition von Symbolen
– Einbeziehung von INCLUDE-Dateien
– Listensteuerung
– Verarbeiten von Code-Makros.

Ausgabeprogramm für Cross-Referenzlisten XREF86

Aufgaben

XREF86 erzeugt aus der vom RASM86 erzeugten Assembler- und Symbolliste eine Cross-Referenzliste.

Bibliothekar LIB86

Aufgaben

Der Bibliothekar ermöglicht dem Nutzer das Erzeugen und Verwalten von Objektbibliotheken, die vom Linker beim Programmverbinden benutzt werden können.

Arbeitsweise

Die einzelnen Funktionen des Bibliothekars sind vom Nutzer über Optionen in der Kommandozeile auswählbar. Folgende Möglichkeiten werden geboten:
– Erzeugen von Objektbibliotheken
– Löschen von Moduln aus einer Bibliothek
– Herauslösen einzelner Moduln aus einer Bibliothek
– Hinzufügen von Moduln zu einer Bibliothek
– Ersetzen von Moduln in einer Bibliothek
– Herstellen einer Liste über die in einer Bibliothek enthaltenen Moduln sowie der darin definierten Segmente und globalen Systeme
– Herstellen einer Cross-Referenzliste für eine Bibliothek.
Der Nutzer kann den Parameterteil für eine LIB86-Operation in eine Datei schreiben und diese Datei beim Aufruf von LIB86 als INPUT-Datei spezifizieren.

Linker LINK86

Aufgaben

Der Linker verbindet einen oder mehrere durch RASM86 oder Compiler für höhere Programmiersprachen erzeugte Objektmoduln zu einem verschieblichen Lademodul, der ab jeder im System möglichen 16-Byte-Grenze ladbar ist.

Arbeitsweise

Der Linker führt folgende Funktionen aus:
– Kombinieren separat erstellter Objektmoduln
– Durchsuchen von Bibliotheksdateien nach Definitionen von benötigten externen Referenzen
– Lösen externer Cross-Referenzen
– Erzeugen einer Ladedatei (CMD-Datei)
– Erzeugen einer Speicherbelegungsliste (MAP-Datei) mit Informationen über die in der CMD-Datei enthaltenen Segmente
– Erzeugen einer Symbolliste (SYM-Datei) mit Informationen über die in der CMD-Datei enthaltenen Symbole.
Der Linkvorgang kann mit der Angabe von Steuerinformationen (Controls) in der LINK86-Kommandozeile beeinflußt werden:
– Controls, die den Aufbau der CMD-Datei beeinflussen
– Controls, die den Aufbau der SYM- und MAP-Dateien beeinflussen

– Controls, die Bibliotheks- und INPUT-Dateien betreffen.
Der Nutzer kann den Parameterteil für eine LINK86-Operation in eine Datei schreiben und diese Datei beim Aufruf von LINK86 als INPUT-Datei spezifizieren.

Debugger SID86

SID86 ist eine Weiterentwicklung des Debuggers DDT86, die es dem Nutzer gestattet, beim Testen mit symbolischen Namen (Variablen-, Prozedur-, Markennamen) zu arbeiten. SID86 benutzt die von LINK86 erzeugte Symbolliste.

6. Die Grafikerweiterung SCP-GX

6.1. Allgemeines

SCP-GX ist eine SCP 1700-Betriebssystemerweiterung, die dem Nutzer eine definierte Grafikschnittstelle bereitstellt. Diese Schnittstelle ermöglicht es dem Nutzer, seine Anwendungsprogramme unabhängig von speziellen Eigenschaften der unterschiedlichen Grafikgeräte zu entwickeln. Für den Aufruf von Grafikfunktionen durch Anwenderprogramme wird eine Konvention benutzt, die der Ruffolge für SCP 1700-BDOS-Funktionen entspricht. Entsprechende Driver für die unterschiedlichen Grafikgeräte übersetzen die SCP-GX-Aufrufe in spezielle Geräteanweisungen. GSX-86, d. h., alle Anwenderprogramme, die GSX-86 benutzen, sind auch mit SCP-GX lauffähig.
Die SCP-Grafikerweiterung besteht in Anlehnung an die SCP-Terminologie aus zwei Teilen – dem Operationssystem für Grafikgeräte GDOS (Graphics Device Operating System) und dem Grafik-E/A-System GIOS (Graphics Input/Output System).

6.2. GDOS

GDOS ist der geräteunabhängige Teil von SCP-GX und erfüllt im wesentlichen drei Aufgaben:

- Realisieren der Schnittstelle zum Anwender, d. h. Entgegennahme von Grafikanforderungen und Rücksenden der Ergebnisse an den Anwender
- Laden von Drivern für Grafikgeräte
- Umformen der vom Anwender erhaltenen normalisierten Gerätekoordinaten (NDC) in Gerätekoordinaten (DC) und umgekehrt.

Der Aufruf von GDOS-Funktionen erfolgt in analoger Weise zu BDOS-Rufen mit dem Softwareinterrupt 224 (Funktionskode 0473H im Register CX) und der Adresse einer Parameterliste in den Registern DX (Offset) und DS (Segment). Die Parameterliste beinhaltet sowohl Ein- als auch Ausgabeparameter.
Die SCP-GX-Funktionen lassen sich in vier Klassen einteilen:

① Ausgabe und Attributsteuerung für Grafik-Primitiven z. B.: Polygon, Polymarke, Füllgebiet, Zellmatrix, Text, verallgemeinerte Darstellungselemente (Balken, Kreisbogen ...). Attribute sind Eigenschaften dieser Primitiven (Farbe, Linienart, Linienstärke, Füllart u. ä.)
② Eingabefunktionen
Lokalisier-, Wertgeber-, Auswähler- und Texteingaben
③ spezielle geräteabhängige Funktionen (z. B. bei Bildschirmgeräten)
Wechsel zwischen Grafik- und Alphanumerikmodus, Kursorsteuerung
④ Steuerfunktionen (z. B. Eröffnen und Abschließen von Grafik-Arbeitsstationen).

Um geräteunabhängig programmieren zu können, werden die Grafikgeräte im Anwenderprogramm mit Identifikationsnummern (ID's) bezeichnet. Anhand einer vom Nutzer zu erstellenden Zuordnungstabelle, die beim Aktivieren von SCP-GX geladen wird, wählt GDOS beim Ruf OPEN WORKSTATION den benötigten Driver aus und lädt ihn, wenn nötig, vom Externspeicher.

6.3. GIOS

Unter GIOS wird im SCP-GX ein Satz von Drivern zum Bedienen der vorhandenen Grafikgeräte verstanden. Die Driver (GIOS-Dateien) müssen als Dateien im SCP 1700-Ladeformat auf einem Externspeicher bereitstehen. Obwohl ein Anwenderprogramm unterschiedliche Grafikgeräte nutzen kann, lädt GDOS immer nur eine GIOS-Datei, wobei die zuvor geladene überlagert wird.
GIOS veranlaßt das Ausführen der im jeweiligen Grafikgerät realisierten grafischen Grundfunktionen. Darüber hinaus werden in bestimmtem Umfang auch SCP-GX-Funktionen emuliert, die ein Grafikgerät mit Hilfe seiner eingebauten Möglichkeiten nicht ohne weiteres ausführen kann.
SCP-GX wird in einer ersten Ausbaustufe mit Drivern für folgende Grafikgeräte bereitgestellt:

- Grafisches Bildschirmsystem des A 7100
- Drucker K 6313
- Plotter K 6418.

Die Dokumentation zu SCP-GX enthält Informationen zum Schreiben eigener Driver.

6.4. Aktivieren der Grafikerweiterung für SCP 1700

SCP-GX wird mit dem Nutzerkommando
ad > GRAPHICS d:
aktiviert, wobei d dasjenige Laufwerk bezeichnet, auf welchem sich die in 6.2. genannte Zuordnungstabelle (ASSIGN.SYS) zwischen Arbeitsstations-ID's und dem zugehörigen Driverdateinamen befindet. Außerdem fungiert d als Standard-Laufwerksnummer für die Dateinamen in ASSIGN.SYS. Ist d nicht angegeben, wird dafür die aktuelle Laufwerksnummer ad eingesetzt.
GRAPHICS lädt SCP-GX zusammen mit dem Driver, der im ASSIGN-SYS an erster Stelle steht, in den Hauptspeicher und initialisiert den Interruptvektor 224 mit dem Eintrittspunkt ins GDOS. GDOS nimmt danach sowohl GDOS- als auch BDOS-Rufe entgegen, wobei letztere direkt ans BDOS weitergeleitet werden.
SCP-GX wird vom Warmstart des Systems (CTRL/C) nicht beeinflußt. Das Deaktivieren der Grafikerweiterung SCP-GX und damit die Freigabe des von ihr belegten Speicherplatzes kann mit dem Kommando ad > GRAPHICS NO angewiesen werden.

Das Timesharing – Betriebssystem MUTOS 1700

Bernd Kubsch
VEB Robotron-Elektronik Dresden

1. Allgemeines

Das MUTOS-Betriebssystem ist ein in vielen Gebieten anwendbares universelles Timesharing-System, das sich aber besonders für die Programmentwicklung, die Dokumentationserstellung, die Lösung wissenschaftlich-technischer und kommerzieller Probleme sowie für die Textverarbeitung eignet. Es ist auch als Basis für Datenbanksysteme, Grafikarbeitsplätze und Auskunftssysteme einsetzbar. MUTOS-Betriebssysteme existieren gegenwärtig für die Rechner K 1630 /1/ und A 5120.16 /4, 12/. Für Rechner-Neuentwicklungen sind gleichartige Systeme vorgesehen.

Im Rahmen des SKR wird von allen Partnerländern arbeitsteilig das Betriebssystem DEMOS (früher: INMOS) für CM4-kompatible Rechner entwickelt.

Für ESER-Betriebssysteme wird mit der PSU (Projektierung und Programmierung stützende Umgebung) eine MUTOS-kompatible Umgebung angeboten /2, 3/.

MUTOS-Betriebssysteme ermöglichen es dem Anwender, auf verschiedenen Rechnerarchitekturen in stets gleicher Weise zu arbeiten. Die Orientierung auf diese Systeme gibt ihm die Gewähr, seine entwickelten Programmpakete mit minimalem Änderungsaufwand auf einer anderen Gerätetechnik nutzen zu können. Außerdem findet er in allen diesen Systemen die gleichen Nutzerschnittstellen vor, sowohl für die Programmierung als auch für die Arbeit mit dem Kommandointerpreter. Programme für MUTOS werden fast ausschließlich in höheren Programmiersprachen geschrieben, vor allem in C. Sie sind oft auch unter Steuerung anderer Betriebssysteme nutzbar, weil alle modernen Entwicklungen C-Compiler enthalten. Ein weiterer Vorteil liegt in seiner leichten Übertragbarkeit auf eine andere Gerätetechnik. Das ist besonders darauf zurückzuführen, daß fast das gesamte Betriebssystem in der Sprache C geschrieben wird. Diese leichte Übertragbarkeit ist ein wesentlicher Faktor dafür, daß internationale für fast alle Rechnerarchitekturen vom Mikro- bis zum Großrechner MUTOS-kompatible Betriebssysteme angeboten werden. Bei Neuentwicklungen wird ein derartiges Betriebssystem meist als erstes angekündigt.

Diese Gründe allein würden jedoch die große Bedeutung nicht rechtfertigen, die MUTOS-kompatible Systeme gegenwärtig haben. Entscheidend trägt dazu die Leistungsfähigkeit des Gesamtsystems in seinen Einsatzgebieten bei. Die Verwirklichung moderner Prinzipien beim Systementwurf, die Leistungsfähigkeit der Sprache C, ein leicht handhabbares, hierarchisches Filesystem, der wohl gegenwärtig leistungsfähigste Kommandointerpreter aller existierenden Betriebssysteme und vor allem ein großes Spektrum leicht verknüpfbarer Programmwerkzeuge haben wesentlichen Anteil an der großen Popularität dieser Betriebssysteme.

MUTOS ist nicht als Echtzeitbetriebssystems für Prozeßsteuerungen geeignet. Das ist vor allem im Fehlen einer prioritätsgesteuerten Prozeßorganisation begründet.

Der vorliegende Beitrag soll zwar speziell das Betriebssystem MUTOS 1700 vorstellen, beschreibt jedoch auch eine große Anzahl von Komponenten, die alle MUTOS-kompatiblen Betriebssysteme enthalten. Eine umfassende Darstellung aller Leistungen dieser Systeme ist in diesem Rahmen unmöglich /5/.

2. Leistungen des Betriebssystems MUTOS 1700

Das Betriebssystem MUTOS 1700 besteht aus den Komponenten
- Kern
- Kommandointerpreter Shell
- Dienstprogramme
- Bibliotheksfunktionen
- Sprachübersetzer und/oder -interpreter.

2.1. Kern

Der Kern umfaßt den hauptspeicherresidenten Teil des Betriebssystems MUTOS 1700. Etwa 90 Prozent dieses Teils sind in der Programmiersprache C geschrieben. Die Hauptaufgaben des Kerns sind:
- Prozeßorganisation
- E/A-Organisation
- Filesystemverwaltung.

Der Kern wird bei der Generierung des Betriebssystems aufgebaut. Über die Angabe bestimmter Parameter ist eine Anpassung des Leistungsumfangs und der Größe des Kerns an konkrete Erfordernisse möglich. Das betrifft zum Beispiel das Einbinden bestimmter Gerätedriver, die Anzahl von Blockpuffern und die Größe interner Tabellen.

Alle Anwenderprogramme (in der MUTOS-Umgebung) werden vom Kern verwaltet. Er sorgt für das Laden abzuarbeitender Prozesse in den Hauptspeicher, für die Vergabe von dynamischen Arbeitsspeicherbereichen, für das Zuteilen von Zeitscheiben an die Prozesse, für die eventuell erforderliche Auslagerung von Prozessen (swapping) und für das Bearbeiten der Systemrufe aus den Prozessen. Über diese Rufe kommuniziert ein Prozeß mit dem Kern des Betriebssystems. Es existieren etwa 50 Systemrufe, die man folgenden Gruppen zuordnen kann:
- Prozeßorganisation
- E/A-Organisation
- Zeitorganisation
- Abfragen und Setzen prozeßspezifischer Informationen
- Signalbehandlung.

Bei der E/A-Organisation ist das gleichartige Behandeln von Files, E/A-Geräten und Pipes besonders erwähnenswert. Auf sehr einfache Art kann eine Umleitung von Ein- oder Ausgaben vorgenommen werden. Es ist problemlos möglich, Daten statt auf ein peripheres Gerät in ein File des Filesystems auszugeben.

Über den Pipe-Mechanismus lassen sich die Ausgangsdaten eines Programms auf den Eingang eines anderen Programms legen. Das ist die Grundlage für das Ketten von Programmen, für das Verknüpfen von Programmwerkzeugen, von denen jedes bestimmte Aufgaben erfüllt. So ist die für MUTOS-kompatible Betriebssysteme typische Philosophie der kombinierbaren,

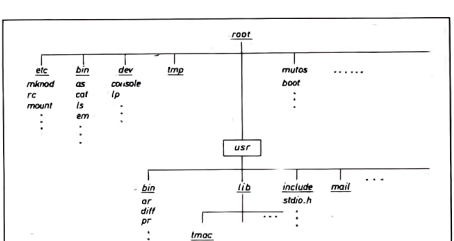


Abb. 1 MUTOS-Filesystem (Ausschnitt)

kleinen Programme mit genau festgelegtem Aufgabenumfang möglich.
Das MUTOS-Filesystem wird direkt vom Kern verwaltet. Sein einfacher, überschaubarer, hierarchischer Aufbau ist einer der Vorzüge des Betriebssystems. Jedes File des Filesystems wird als kontinuierliche Folge von Bytes betrachtet, die einzeln adressierbar sind. Für das Betriebssystem hat der Inhalt der Files keine bestimmte Struktur. Sie wird nur von den verarbeitenden Programmen festgelegt. So ist es möglich, auf der Basis dieses einfachen Filesystems spezielle Organisationsformen (z. B. Zugriffsmethoden) aufzubauen.
Zum Beschleunigen des Filezugriffs erfolgt eine Pufferung von Blöcken des Filesystems im Kern des Betriebssystems. Daten werden nicht sofort auf den Datenträger übertragen, sondern in den Blockpuffern gespeichert, solange noch freie Puffer vorhanden sind. Damit können weitere Zugriffe auf Daten dieses Blockes ohne physische Übertragung vom Externspeicher und somit sehr zeiteffektiv erfolgen. Zu den Nachteilen dieses Verfahrens gehört jedoch die viel kritisierte Anfälligkeit des Filesystems gegen E/A-Störungen seitens der Hardware und gegen unkontrolliertes Abschalten des Systems. Ein automatisches, zeitzyklisches Aktualisieren der Filesysteme (update) reduziert die Probleme.
Das Filesystem hat einen baumartigen Aufbau. Es enthält Directories (Verzeichnisse), reguläre Files und Spezial-Files. Jedes Directory kann wiederum Directories enthalten, so daß sich, ausgehend von einer Wurzel (Root-Directory), die Baumstruktur ergibt. An ein Directory kann über den mount-Systemruf ein anderes Filesystem angebunden werden. Damit kann ein Filesystem um eine beliebige Anzahl von Datenträgern (Filesystemen) erweitert werden.
Reguläre Files enthalten Daten. Die Spezial-Files sind Directory-Eintragungen zum Ansprechen peripherer Geräte. Filenamen können aus bis zu 14 alphanumerischen und Sonderzeichen bestehen. Die Folge von Directory-Namen bis zum jeweiligen File wird als Pfadname bezeichnet. Die einzelnen Teile des Pfades sind durch "/" getrennt. Das Root-Directory wird durch ein "/" am Beginn des Pfadnamens gekennzeichnet.
Die Abbildung 1 zeigt einen Ausschnitt aus einem MUTOS-Filesystem. An das Directory /usr des Root Filesystems ist ein weiteres Filesystem angebunden. Beispiele für Pfadnamen:
/usr/lib/tmac/tmac.s für eine File, das vom Textformatierungsprogramm nroff genutzt wird, und
/dev/console für das Spezial-File zum Ansprechen der Bedieneinheit.
Für jedes Directory, reguläre File und Spezial-File gibt es bestimmte Zugriffsrechte. Sie betreffen das Lesen, Schreiben und Abarbeiten (bei Directories das Durchsuchen) und werden für den Eigentümer des Files, die anderen Mitglieder der Nutzergruppe des Eigentümers und den Rest der Nutzer getrennt vergeben. Bei allen Zugriffen auf Files erfolgt ein Überprüfen der Zugriffsrechte des auslösenden Nutzers. So ist ein recht guter Schutz des einzelnen Files gegen unberechtigte Benutzung gegeben.

2.2. Kommandointerpreter Shell

Der im Abschnitt 2.1. beschriebene Kern und seine Leistungen sind vom Nutzer nicht direkt ansprechbar. Dies ist nur Programmen möglich. Bevor der Anwender sich solche Programme geschrieben hat, übernimmt der Kommandointerpreter die Kommunikation. Wie der Name Shell (Schale) ausdrückt, legt sich dieses Programm um den Kern des Betriebssystems. Alle MUTOS-Leistungen werden von der Shell mit einer sehr mächtigen Kommandosprache nutzbar gemacht. Es sei jedoch betont, daß die Shell für MUTOS ein Programm wie jedes andere ist und problemlos gegen einen nutzerspezifischen Kommandointerpreter ausgetauscht werden kann.
Shell-Anweisungen stellen sowohl eine Kommando- als auch eine Programmiersprache dar. Die Shell liest die Kommandos vom Terminal oder aus einem File und führt sie aus. Das Abspeichern von Prozeduren in Files ermöglicht es, eigene Kommandos zu bilden und diese wie Systemkommandos zu benutzen.
Die Möglichkeiten der Shell sollen an einigen wesentlichen Funktionen dargelegt werden. Einfache Shell-Kommandos bestehen aus einer Folge von Worten, die durch Leerzeichen oder Tabulatoren getrennt sind. Das erste Wort ist der Name des Kommandos, das ausgeführt werden soll. Alle restlichen Worte werden dem auszuführenden Kommando als Parameter übergeben. So listet das Kommando
ls -l
die Namen der Files im laufenden Directory aus. Dabei wertet ls das Argument -l (für "long") aus und zeigt außer den Namen der Files weitere Informationen an.
Diese einfache Form von Shell-Kommandos kann von Anweisungen ergänzt werden (dargestellt durch bestimmte Sonderzeichen), die von der Shell ausgewertet und nicht an das aufgerufene Kommando übergeben werden. Mit dem Anhängen eines "&" an eine Kom-

mandozeile erreicht man, daß das ausgelöste Kommando im Hintergrund abgearbeitet wird. Die Shell wartet dann nicht auf den Abschluß des Kommandos, wie das sonst üblich ist.
Für jeden Prozeß eröffnet die Shell drei Standardfiles. Sie werden als Standard-Eingabefile*, Standard-Ausgabefile** und Standard-Fehlerausgabe*** bezeichnet.
Normalerweise sind sie alle drei dem auslösenden Terminal zugewiesen. In der Kommandozeile ist eine E/A-Umweisung möglich. Mit " " wird das Standard-Ausgabefile, mit " " das Standard-Eingabefile auf das dahinter angegebene File oder Spezial-File umgewiesen. Die Schreibweise " " bewirkt das Anhängen an ein File.
ls -l file1
legt das Ergebnis des ls-Laufes im File file1 ab,
ls -l file2
schreibt es an das Ende des Files file2.
wc file3 file4
ermittelt die Anzahl von Zeichen, Worten und Zeilen des Files file3 und gibt das Ergebnis in das File file4 aus.
Eine sehr nützliche Eigenschaft des Betriebssystems und speziell der Shell ist die Möglichkeit, das Standard-Ausgabefile eines Kommandos mit dem Standard-Eingabefile eines anderen Kommandos zu verbinden. So können Daten ohne Zwischenspeicherung übergeben werden. Dieses Prinzip wird Pipelining genannt. Als Pipe-Operator ist "!" anzugeben.
ls -l ! wc
hat das gleiche Resultat wie
ls -l file 5; wc file5 ; rm file5
Das Pipe-Prinzip ist eine wesentliche Voraussetzung für die Verknüpfbarkeit von Prozessen und damit von Programmwerkzeugen, von denen jedes bestimmte, abgegrenzte Funktionen ausführt. Viele der MUTOS-Kommandos lesen vom Standard-Eingabefile und geben das Ergebnis auf das Standard-Ausgabefile. Sie wirken als Filter. Das geschickte Verknüpfen mehrerer derartiger Filter erspart in vielen Fällen eigene Programmierarbeiten.
ls ! grep new ! sort ! pr -3 /dev/lp
erzeugt eine dreispaltige, sortierte Liste aller Files des laufenden Directory, deren Namen die Zeichenkette new enthält, und gibt sie auf dem Drucker aus.

* (standard input)
** (standard output)
*** (standard error)

Der Aufbau von Prozeduren wird mit einer Reihe von Anweisungen zur Ablaufsteuerung unterstützt. Sie ähneln denen von höheren Programmiersprachen:

for ... do ... done
for ... in ... do ... done
while ... do ... done
until ... do ... done
case ... in ... esac
if ... then ... elif ... else ... fi

Shell bietet weiterhin umfangreiche Möglichkeiten der Zeichenketten-, Parameter- und Kommandosubstitution, der Arbeit mit Variablen, der Bildung von Programmgruppen, des Tests von Prozeduren, der Fehlerbehandlung sowie eine Reihe von Kommandos, die von der Shell selbst ausgeführt werden, ohne eines der sonstigen MUTOS-Kommandos zu aktivieren.
Am folgenden Beispiel soll ein Eindruck vom Arbeiten mit den Shell-Funktionen vermittelt werden. Um die Erklärungen besser zuordnen zu können, wurden die Zeilen der Prozedur numeriert, was sonst nicht der Fall ist.
Das Kommando soll dirlist heißen und den Inhalt eines als Parameter angegebenen Directory sowie eventuell vorhandener Subdirectories ausgeben. Über eine Option -l läßt sich eine Ausgabe im long-Format anweisen:

```
 1: old='pwd'
 2: lflag=
 3: d='pwd'
 4: for i
 5: do
 6: case $i in
 7: -l) lflag="-1";;
 8: -*) echo "Benutzung : dirlist -l directory";;
 9:  *) if test -d $i
10: then d=$i
11: else
12: echo "$i ist kein Directory"
13: fi
14: esac
15: done
16: cd $d
17: echo"
18: • • •Inhalt des Directory 'pwd'• • •
19: "
20: ls $lflag
21: for i in*
22: do
23: if test -d $i
24: then dirlist $lflag $i
25: fi
26: done
27: cd $old
```

Beschreibung der Zeilen:

1	Der Name des laufenden Directory wird der Shellvariablen old zugewiesen (Kommandosubstitution, Nutzung des MUTOS-Kommandos pwd (print working directory))
2	lflag soll die Option -l enthalten, falls diese beim Aufruf angegeben wird (Standard: ohne -l)
3	Wird kein Directory beim Aufruf von dirlist angegeben, soll das laufende Directory benutzt werden
4...15	Analyse der Kommandozeile
4	Für sämtliche Parameter des Aufrufes wird geprüft:
7	wurde -l angegeben? ja, Setzen von lflag
8	wurde eine andere Option angegeben? ja, Fehlermeldung
9	wurde ein Directory angegeben?
10	ja, Verändern der Shellvariablen d
12	nein, Fehlermeldung
16	Setzen des Arbeitsdirectory auf das auszulistende Directory
18...19	Ausgabe einer Kopfzeile
20	Ausgabe des Directory-Inhalts mit dem MUTOS-Kommando ls (Kurz- oder Langform)
21...26	Für alle Files mit auszulistenden Directory wird geprüft:
/23/	ist das File ein Directory? ja, rekursiver Aufruf von dirlist
27	Wiederherstellung des ursprünglichen Arbeitsdirectory.

Das Ergebnis eines Aufrufes von dirlist kann folgendermaßen aussehen:

```
• • •Inhalt des Directory /usr/peter• • •
total 4
-rw-r--r-- 1 peter      24 Apr 3 10:56 file1
-rwxrwxr-x 1 peter     157 Apr 3 10:56 file2
drwxrwxr-x 2 peter      32 Apr 3 10:48 tmp
```

```
drwxrwxr-x 2 peter     96 Apr  3 11:09 work
• • •Inhalt des Directory /usr/peter/tmp• • •
total 0
• • •Inhalt des Directory /usr/peter/
work• • •
total 3
-rwxr-xr-x 1 peter    378 Apr  3 10:59 dl
-rw-rw-r-- 1 peter    450 Apr  3 11:02 dl.sh
-rw-rw-r-- 1 peter    332 Apr  3 11:13 liste
```

Die umfangreichen Leistungen der Shell und der große Vorrat an MUTOS-Programmwerkzeugen ermöglichen es, in vielen Fällen ohne das Schreiben spezieller Programme zu befriedigenden Problemlösungen zu kommen.

2.3.
Sprachübersetzer- und/oder -interpreter

Die für MUTOS 1700 vorgesehene Unterstützung höherer Programmiersprachen ist im Abschnitt Sprachkonzeption des A 7100 dargelegt. An dieser Stelle sollen dazu nur einige Ergänzungen erfolgen.

Der unmittelbar mit dem Betriebssystem MUTOS 1700 verknüpfte Compiler ist der für die Sprache C. Er entspricht dem C-Compiler des MUTOS 1630.

Zum MUTOS 1700 gehört auch ein Assembler, der jedoch für die Entwicklung von Anwenderprogrammen nicht zu empfehlen ist. Seine Hauptanwendung liegt im Übersetzen des Assemblerzwischencodes (den der C-Compiler erzeugt) in die Maschinensprache. C- und Assemblermodule zu kombinieren ist unproblematisch. Eine Kompatibilität zu anderen Assemblern besteht nicht.

In die Gruppe der Sprachübersetzer fallen einige Programme, die sehr nützliche Hilfen beim Schreiben von Programmen sind:

- lint ist ein Programm zur lexikalischen Analyse von C-Programmen. Diese Analyse ist umfassender als sie vom C-Compiler vorgenommen wird. Lint kennzeichnet auch solche Konstruktionen als problematisch, die von der Sprache zwar zugelassen sind, jedoch Fehlerquellen sein können. Sie können auf diesem Wege gefunden und vermieden werden.
- m4 ist ein Makroprozessor, der außer dem üblichen Ersetzen eines Textes durch einen anderen auch Makros mit Argumenten, bedingte Makroexpansionen, arithmetische Ausdrücke, Filemanipulationen und spezielle Zeichenkettenverarbeitung unterstützt.
- yacc ist ein Programmgenerator, der auf der Basis einer vorgegebenen Grammatik ein C-Programm zur Syntaxanalyse erstellt. Dabei besteht die Möglichkeit, Aktionen für die statistische und dynamische Semantik in die Grammatik einzubinden.

Schließlich kann auch awk zu diesen Hilfsmitteln gezählt werden (siehe 2.5.4.).

Die genannten Programme erlauben es, auf eine Software-Entwicklung auf Assemblerniveau völlig zu verzichten. Die Effektivität des Entwicklungsvorganges, die Qualität und Portabilität der Ergebnisse sind damit gegenüber anderen Klein- und Mikrorechner-Betriebssystemen deutlich zu verbessern.

2.4.
Bibliotheken

Das Betriebssystem MUTOS 1700 verfügt im wesentlichen über zwei Bibliotheken – die Systembibliothek libc.a und die Bibliothek mathematischer Funktionen libm.a.

2.4.1.
Systembibliothek libc.a

Die Systembibliothek libc.a enthält alle Eintrittspunkte zu den Basisfunktionen des Betriebssystemkerns (Systemrufe) und zu einer Reihe weiterer Funktionen, die auf den Systemrufen aufbauen. Sie bieten dem Anwender mehr Komfort beim Ausnutzen weiterer Systemleistungen. Alle Funktionen sind mit dem Ziel einer möglichst hohen Zeit- und Speicherplatzeffektivität geschrieben worden und garantieren auf Grund ihrer Maschinenunabhängigkeit die Portabilität zu anderen Rechnersystemen. Darüber hinaus können Programme, die nur die Standard-Funktionen verwenden, auf jedem Rechner mit MUTOS-Umgebung abgearbeitet werden.

Kernstück der Systembibliothek ist das Standard-E/A-Paket. Es verfügt neben Funktionen für die unformatierte Ein- und Ausgabe einzelner Zeichen, Worte oder Zeichenketten auch über Funktionen für die formatierte Ein-/Ausgabe. Diese sind sehr vielseitig. Sie wandeln z. B. numerische Werte in Zeichenketten um und umgekehrt. Die Quelle und das Ziel jeder E/A-Operation wird von einem Filepointer beschrieben. Dieser wird mit Hilfe der Funktion fopen, die als Argumente einen Filenamen und eine Beschreibung der gewünschten Zugriffsrechte (Lesen und/oder Schreiben) verlangt, geliefert. Alle weiteren E/A-Operationen werden auf der Basis dieses Filepointers durchgeführt.

Die Systembibliothek liefert aber nicht nur für die E/A-Funktionen die Systemschnittstelle, sondern für alle in Form von Systemrufen verfügbaren Leistungen des Betriebssystemkerns und umfangreiche Hilfsfunktionen. Dazu gehören Leistungen zum Unterstützen
– der Zeichen- und Zeichenkettenverarbeitung
– der dynamischen Speicherverwaltung
– der Prozeßsteuerung
– der Arbeit mit Filesystemen und
– der Verwendung von Datum und Uhrzeit.

2.4.2.
Mathematische Bibliothek libm.a

Die mathematischen Funktionen sind in der Bibliothek libm.a enthalten. Neben den trigonometrischen, hyperbolischen und den Exponentialfunktionen sind auch verschiedene Besselfunktionen realisiert. Während die Standardbibliothek vom C-Compiler automatisch geladen wird, ist die mathematische Bibliothek mit einer Kommandooption explizit zum Durchsuchen nach benötigten Funktionen bereitzustellen.

2.5.
Dienstprogramme

2.5.1.
Überblick

MUTOS enthält ein umfangreiches Spektrum an Dienstprogrammen (Systemprogramme). Es ist hier nicht möglich, auf alle näher einzugehen. Sie werden deshalb zu Programmgruppen zusammengefaßt, und nur einige wenige werden ausführlicher beschrieben. Der Schwerpunkt liegt dabei auf den Mit-

teln zur Programmentwicklung und zur Textverarbeitung. Dienstprogramme gibt es für folgende Aufgabenkomplexe:

- Filemanipulation
- Manipulation von Directories und Filenamen
- Abarbeiten von Nutzerprogrammen
- Kommunikation zwischen Nutzern
- Programmentwicklung
- Arbeit mit den implementierten Programmiersprachen
- Textbearbeitung
- Textverarbeitung und Dokumentationserstellung
- Statusabfragen, Wartung und Betrieb des Systems.

Zunächst werden die Programmgruppen mit einigen typischen Vertretern und deren Funktionen aufgezählt. Im Anschluß daran wird etwas ausführlicher auf die Schwerpunkte

- em
- awk
- nroff, tbl, neqn

eingegangen.

Em ist der von MUTOS 1630 her bekannte, leistungsfähige Bildschirmeditor. Genaueres über ihn kann sonst nur in /1/ nachgelesen werden.

Awk wurde ausgewählt, da der Interpreter dieser gleichnamigen Mustererkennungs- und Musterverarbeitungssprache relativ selten benutzt wird, jedoch vielfältig bei Textmanipulationen eingesetzt werden kann.

Die Komponenten nroff, tbl und neqn sollen als Werkzeuge zur oft benötigten Dokumentationserstellung hervorgehoben werden.

2.5.2. Programmgruppen

Filemanipulation:

cp Kopieren
cptree Kopieren von vollständigen Filehierarchien
cmp Vergleich zweier Files
cat Verketten von Files, auch Einzelausgabe
split Trennen von Files
tail Ausgabe von Files ab bestimmter Position
dd Übertragen von Files und Umsetzen von Fileformaten
sum Prüfsummenbildung

Manipulation von Directories und Filenamen:

mkdir Anlegen eines neuen Directory
rmdir Löschen eines Directory
ln Erzeugen von Links (Alias-Namen)
rm Löschen von Files und Directory-Hierarchien
mv Umbenennen von Files
chmod Ändern von Zugriffsrechten zu Files und Directories
chown Ändern des Eigentümers von Files und Directories
chgrp Ändern der Nutzergruppe für Files und Directories
cd Ändern des aktuellen Directory eines Nutzers
find Durchsuchen von Directory-Hierarchien

Abarbeitung von Nutzerprogrammen:

sh Kommandointerpreter Shell
test Test von Files nach bestimmten Kriterien
expr Ermitteln von Kommandoargumenten
wait Warten auf die Beendigung asynchron laufender Prozesse
sleep Verzögern einer Kommandoausführung
nice Verändern der Priorität eines Kommandos
kill Abbrechen eines Prozesses

Kommunikation zwischen Nutzern:

mail Hinterlassen einer Nachricht für einen Nutzer
write Direkte Kommunikation zwischen Nutzern
wall Übertragen einer Nachricht an alle Nutzer
mesg Unterdrücken des Empfangs von Nachrichten

Programmentwicklung:

ar Archivierung von Files in Bibliotheken
adb Interaktives Testsystem
od Ausgabe beliebiger Files in verschiedenen Formaten
ld Lader für Objektprogramme
lorder Ordnen von Objektprogrammbibliotheken zum Laden
nm Entfernen der Symboltabelle von Objektprogrammen
strip Entfernen der Symboltabelle und der Verschiebeinformationen
size Ausgabe der Speicherplatzanforderungen von Objekt-Files
time Ermitteln von Laufzeiten bei der Kommandoabarbeitung
prof Statistische Angaben über Ausführungszeiten in Programmen
make Steuerung beim Erstellen großer Programme

Arbeit mit den implementierten Programmiersprachen:

cc C-Compiler entsprechend UNIX, Version 7
cb Formatierungsprogramm für C-Programme
as Assembler
m4 Makroprozessor mit breitem Anwendungsgebiet
lint Syntax- und Plausibilitätsprüfungen für C
yacc Programmgenerator für die Syntax-Analyse

Textbearbeitung:

ed Interaktiver Texteditor (Kommandomodus)
em Erweiterter Editor mit Bildschirmmodus
sed Stream-Editor für vorprogrammierte Arbeitsweise
sort Zeilenweises Sortieren von Text-Files
tsort Sortieren nach einem vorgegebenen Muster
unig Aussondern mehrfach aufeinanderfolgender Zeilen
prep Statistische Aufbereitung von Texten
rev Umkehren von Textzeilen in Files
tr Umcodieren von Zeichen nach einem beliebigen Schlüssel
diff Ermitteln aller Unterschiede zweier Text-Files
comm Durchsuchen zweier sortierter Files nach gleichen Zeilen
grep Ermitteln aller Zeilen eines Files, die eine vorgegebene Zeichenkette enthalten
wc Zählen von Zeilen, Worten und Zeichen eines Text-Files
awk Mustererkennungs- und Verarbeitungsprogramm

Textverarbeitung und Dokumentationserstellung:

look Suchen nach Worten in einem sortierten File (Wörterbuch)
nroff Textformatierungsprogramm
neqn Formatierungsprogramm zum Setzen mathematischer Gleichungen und Formeln
tbl Preprozessor für nroff zum Tabellenaufbau

Statusabfragen, Wartung und Betrieb des Systems:

ls	Auslisten von Filenamen und zugehörigem Status
file	Ermitteln des Typs eines Files
date	Ausgeben bzw. Setzen von Datum und Uhrzeit
df	Ermitteln der freien Blöcke von Filesystemen
du	Ermitteln der Anzahl von Blöcken, die von Files belegt sind
quot	Ermitteln der Anzahl von Blöcken, die jeder Nutzer besitzt
ps	Ausgabe von Informationen über laufende Prozesse
pstat	Ausgabe von Informationen über den Systemzustand
iostat	Ausgabe statistischer Informationen über E/A-Aktivitäten
pwd	Ausgabe des aktuellen Directory-Namens
dir	Ausgabe des Directory-Inhalts
mount	Eingliedern von Filesystemen
umount	Ausgliedern von Filesystemen
mkfs	Einrichten eines neuen Filesystems auf einem Datenträger
mknod	Einrichten eines Spezial-Files
tar	File-Archivierung auf Datenträgern in speziellem Format
dump	Abzug eines Filesystems auf einen Datenträger
restor	Rückspeichern eines dump-Abzuges
fsck	Prüfen und Reparieren von Filesystemen

2.5.3. Editor em

MUTOS bietet drei Editoren an – die interaktiven Texteditoren ed und em sowie den Stream-Editor sed.
Der Stream-Editor eignet sich gut, wenn viele möglichst vorprogrammierte Schritte in einem File auszuführen sind. Er eignet sich auch zum Bearbeiten sehr großer Files.
Ed und em sind für die interaktive Arbeit vorgesehen. Sie können nur Files verarbeiten, deren Zeilenzahl den Datenbereich des Editorprogramms nicht überschreitet. Em ist der interessantere und häufiger genutzte Editor. Seine Leistungsfähigkeit ist beeindruckend. Die Funktionen des ed sind als Untermenge vorhanden. Er hat zwei Arbeitsmodi: den Kommandomodus und den Bildschirmmodus. Im Kommandomodus werden die auszulösenden Funktionen über Zeichenketten eingegeben, während im Bildschirmmodus hauptsächlich Funktionstasten benutzt werden. Einige der Kommandos aus dem Normalmodus sind auch im Bildschirmmodus vorhanden. Die wesentlichen Vorteile des Bildschirmmodus liegen in der bequemen Arbeitsweise beim Aufbereiten von Texten. Dabei werden die Tasten zur Cursorbewegung sowie alle existierenden Funktionstasten für das Positionieren im Text, für das Löschen, Überschreiben und Einfügen von Zeichen und Zeilen, für das Trennen und Verbinden von Zeilen, für das Vorwärts- und Rückwärtsblättern und für spezielle Funktionen genutzt. Eine Darstellung des Funktionsangebots auf den untersten Bildschirmzeilen erleichtert das Bedienen.
Textzeilen können sich über mehrere Bildschirmzeilen erstrecken. Eine weitere Besonderheit des em ist das Führen eines log-Files über alle im Verlauf der Editorarbeit erfolgten Schritte. Bei nicht ordnungsgemäßem Beenden der Arbeit (z. B. bei einem Systemausfall) ist die Gewähr gegeben, daß alle Schritte bis zum Abbruch automatisch wiederholt werden können. Diese Recovering-Funktion vermeidet, daß längere Aufbereitungen vergeblich waren.

2.5.4. Mustererkennungs- und Musterverarbeitungssprache awk

Das Dienstprogramm awk ist ein Interpreter für die Mustererkennungs- und Musterverarbeitungssprache awk. Mit Hilfe dieser Sprache werden Programme formuliert, die in einem oder mehreren Files nach Zeilen mit bestimmten Textmustern suchen und in Abhängigkeit des Resultats dieser Suchprozesse die ebenfalls formulierten Aktionen ausführen. Im allgemeinen besteht ein awk-Programm aus Anweisungen der Form

Textmuster Aktion

Jede Eingabezeile eines Files wird der Reihe nach mit jedem der vorgegebenen Textmuster verglichen. Bei einem erfolgreichen Vergleich wird das Ausführen der dazugehörigen Aktion veranlaßt. Dabei kann sowohl das Textmuster fehlen – die Aktion wird dann für jede Zeile ausgeführt – als auch die Aktion. In diesem Fall wird die Zeile einfach in das Standard-Ausgabefile kopiert. Das einfachste awk-Programm besteht aus der Aktion

print

und veranlaßt das Herstellen einer Kopie des im Kommandoaufruf angegebenen Files.
Das Textmuster kann aus einer Vielzahl logisch miteinander verknüpfter Ausdrücke bestehen. Die Aktionen können ebenfalls wieder komplizierte Verarbeitungen durchführen. Es sind z. B. solche Sprachelemente wie if-Anweisungen, while- und for-Schleifen möglich. Dadurch wird eine sehr große Flexibilität erreicht. Als Beispiel für das Anwenden von awk wird die formatierte Ausgabe eines Wörterbuches Englisch-Deutsch-Russisch vorgestellt. Das Input-File enthält Textzeilen folgender Form:

```
...
computer:Rechner:vychislitelnaia mashina
⋮
file:Datei:fajl
⋮
input:Eingabe:vvod
⋮
output:Ausgabe:vyvod
...
```

Jede Eingabezeile wird vom awk-Interpreter als Satz betrachtet. Der Satztrenner kann beliebig sein (in diesem Beispiel ist es das Standard-Zeichen für Zeilenende). Als Feldbegrenzung des Eingabetextes FS wird der Doppelpunkt definiert – für den Ausgabetext wird ein Tabulator festgelegt (OFS). Je nach Wunsch kann nun ein beliebiges, alphabetisch sortiertes Wörterbuch ausgegeben werden.
Englisch-Deutsch:

```
BEGIN FS=":"; OFS=" "
printf "%-50s%-50s n", $1, $2
```

Falls nur die mit den Buchstaben "i" und "o" beginnenden Wörter gewünscht werden:

```
BEGIN FS=":"; OFS=" "
/$1 ~ i66$1 ~ o/ printf "...", $1, $2
```

Das entsprechende awk-Programm wird im File prog abgelegt. Zunächst muß aber alphabetisch sortiert werden (falls es nicht schon aus zeiteffektiven Gründen sortiert abgelegt worden ist), ehe das Ausgabeformat aufbereitet werden kann:

```
sort -fdt: +0 -1 woerterbuch !awk -f prog
!pr
```

Deutsch Russisch-Englisch:

```
BEGIN FS= ":"; OFS=" "
printf "%-40s%-40s%-40s n", $2, $3, $1
sort -fdt: +1 -2 woerterbuch !awk -f prog
!pr
```

2.5.5. Dokumentationsherstellung

Ein sehr wesentlicher, aber auch sehr aufwendiger und unbeliebter Teil aller Programmieraufgaben besteht in der Dokumentationsherstellung. Aber auch im Büro, in den Redaktionen und Druckereien, ja eigentlich in allen Zweigen der Volkswirtschaft fallen umfangreiche Textverarbeitungsaufgaben an. MUTOS bietet eine Vielzahl von Werkzeugen zur Rationalisierung dieser aufwendigen Arbeiten. Die wichtigsten sind:

nroff – ein Textformatierungsprogramm zur vollautomatischen Herstellung vielgestaltiger Texte und kompletter Dokumentationen

tbl – ein Preprozessor für nroff zur Tabellenformatierung

negn – ein Preprozessor für nroff zur Formatierung mathematischer Ausdrücke.

Der Editor als Grundvoraussetzung jeglicher redaktionellen Bearbeitung wurde bereits im Abschnitt 2.5.3. vorgestellt. Die folgenden Ausführungen sollen lediglich zum Verständnis der Leistungsfähigkeit der MUTOS-Textformatierung beitragen. Umfassende Erläuterungen, Anwendungsbeispiele und Bedienerhinweise sind in/1/ enthalten.

Textformatierung mit nroff

Das Textformatierungsprogramm nroff arbeitet auf der Basis von in den Text eingebauten Kommandos für die Formatierung und Steuerfolgen zur Gerätesteuerung bei der Ausgabe. In der Regel sind das Steuerzeichen für Drucker oder Bildschirmausgabegeräte. In naher Zukunft werden aber die wesentlich flexibleren und eine qualitativ bessere Ausgabe liefernden Lichtsatzgeräte unterstützt werden können.

Dem Anwender werden zur Erleichterung der Formatierungs- und Ausgabesteuerungen umfangreiche Makropakete angeboten. Das heißt, der Eingabetext enthält das Manuskript mit Steuerzeilen, die die Formatierung des Textes in der gewünschten Form veranlassen. Diese Steuerzeilen enthalten die Makrokommandos, die von nroff aufgelöst und abgearbeitet werden. Zum besseren Verständnis wird im folgenden eine Auswahl von Makrokommandos aufgezählt:

.2C – Ausgabe des Textes im Zweispaltenformat
.B – Ausgabe im Fettdruck (Schattenschrift)
.I, .R – Ausgabe im Kursivdruck
.OP – Ausgabe im Normaldruck
.IP – Ausgabe im Mehrfachdruck (Überdruck)
.PP – Beginn eines eingerückten Paragraphen
.RP – Beginn eines neuen Paragraphen, nur die 1. Zeile ist eingerückt
– Ausgabe eines Deckblattes.

Die Parameter der Makrokommandos enthalten Angaben darüber, welche Textstücke wie zu behandeln sind. Die gerätespezifischen Besonderheiten sind in einer Gerätetabelle enthalten. So werden z. B. die Umlaute im nroff-Programm als eine genau definierte konstante Zeichenfolge behandelt. Erst in der Gerätetabelle wird diese Zeichenfolge entschlüsselt und entweder als ae (z. B. bei Bildschirmausgaben) oder als a backspace" ausgegeben. Je nach Drucker sind verschiedene Ausgabe-Zeichenfolgen möglich. Die Namen sowohl des Makropakets als auch der Gerätetabelle sind beim Aufruf von nroff anzugeben.

Der Formatierer bearbeitet Folgen von druckbaren Zeichen (Worte), die von mindestens einem Leerzeichen getrennt sind. Unter Berücksichtigung der Vorgaben für den rechten und linken Rand werden so viele Worte wie möglich in die Ausgabezeile aufgenommen. Das nroff-Programm des MUTOS realisiert die Silbentrennung nach den Regeln der deutschen Grammatik. Das heißt, es wird versucht, das letzte Wort zu trennen, falls es über den rechten Rand hinausgeht. In jedem Fall kann aber ein Randausgleich veranlaßt werden.

```
.TS
center allbox;
c s
l l
c c.
W/orterbuch
Deutsch Englisch
Auto    car
Baum    tree
Wort    word
.TE
```

Wörterbuch	
Deutsch	Englisch
Auto	car
Baum	tree
Wort	word

Abb. 2 Beispiel zum Verarbeiten der Kommandos durch den Preprozessor

Tabellenformatisierung mit tbl

Das Dienstprogramm tbl dient der automatischen Generierung von Tabellen, die in beliebigen, von nroff zu verarbeitenden Text eingeschoben werden können. Die Eingabe der Tabelle erfolgt ebenfalls unformatiert. Eingeschobene Kommandos steuern den Ausgabestrom, der dann wiederum von nroff weiterverarbeitet wird. Für den Aufruf der Kommandofolge kann deshalb der Pipe-Mechanismus angewendet werden:

```
$tbl textundtabelle !nroff
```

Der Preprozessor tbl verarbeitet alle zwischen den Kommandos .TS (Tabelle Start) und .TE (Tabelle Ende) eingeschlossenen Zeilen. Zum besseren Verständnis siehe Beispiel in Abb. 2.

Formatierung mathematischer Ausdrücke mit neqn

Das Dienstprogramm neqn arbeitet wie tbl als Preprozessor für nroff. Es wer-

den alle zwischen den Kommandos .EQ und .EN eingeschlossenen Zeilen verarbeitet. Die mathematischen Ausdrücke werden innerhalb dieser Begrenzung mit Hilfe von Schlüsselworten beschrieben. Folgende Beispiele seien angeführt:

sup – hochgestellte Indizes
sub – tiefgestellte Indizes
over – Brüche
sgrt – Quadratwurzeln
matrix – Matrizen.
So ergibt der Ausdruck
1 over sgrt ax sup 2 +bx+c
die Darstellung

$$\frac{1}{\sqrt{ax^2 + bx + c}}$$

3.
Arbeitsweise auf dem A 7100
3.1.
Gerätekonfiguration

Das Betriebssystem MUTOS 1700 setzt folgende Minimalkonfiguration voraus:
– Arbeitsplatzcomputer A 7100 mit den Bestandteilen
Prozessor K 1810 WM86
programmierbare Interruptsteuereinheit K 850 WN59A
programmierbarer Intervallzeitgeber K 580 WI53
256 KByte Operativspeicher (RAM)
zwei Minifolienspeicherlaufwerke K 5600.20
eine Bildschirmeinheit K 7229
eine Tastatur K 7637
Zusätzlich können bedient werden:
– voller Ausbau des Operativspeichers
– zwei Minifolienspeicherlaufwerke K 5600.20
oder zwei Standard-Folienspeicherlaufwerke (in einem Beistellgefäß) K 5602.10
– ein ASP mit den Interfaces
ein IFSS
ein IFSP
eine V.24.

3.2.
Aufteilung der Betriebssystemkomponenten auf die Disketten

Die Kapazität der Disketten reicht nicht aus, um alle zum System gehörenden Komponenten ständig im direkten Zugriff zu halten. Deshalb erfolgt eine Aufteilung der Systemprogramme und Bibliotheken auf mehrere Filesysteme (Disketten), die je nach Bedarf über das Kommando mount an das Root-Filesystem angebunden werden können. Die Aufteilung ist nicht starr, sondern kann in weiten Grenzen vom Anwender selbst vorgenommen werden. Sinnvoll erscheint eine Gruppierung nach Aufgabengebieten. Eine Diskette wird als Systemdiskette (Root-Filesystem) die ständig oder häufig benötigten Kommandos sowie den Auslagerungsbereich (swap space) enthalten. Auf ihr können auch die temporären Files einiger Programme und der Pipe-Organisation angelegt werden. Ein weiterer Komplex wird die Komponenten umfassen, die für die Programmentwicklungen benötigt werden. Dazu gehören z. B. Editor, Compiler, Assembler, Lader, Archivar, Debugger, make und Bibliotheken. Die Komponenten der Textverarbeitung (Editor, awk, nroff, tbl, neqn, grep, wc u. ä. sowie zugehörige Bibliotheken) bilden ebenfalls eine aufgabenspezifische Gruppe.
Nach der Häufigkeit der Benutzung kann sich der Anwender diese Filesysteme aus dem vollen Spektrum der MUTOS-Komponenten so zusammenstellen, wie es seinen Arbeitsbedingungen entspricht.

3.3.
Arbeitsweise des Kerns

Die zunächst geringe Externspeicherkapazität und der relativ langsame Zugriff auf die Disketten schließen in den meisten Fällen einen Mehrnutzerbetrieb aus, der jedoch prinzipiell möglich ist. Außerdem besteht die Gefahr, der gegenseitigen Störung im Mehrnutzerbetrieb auf Grund von Programmierfehlern, solange man über keine Speicherverwaltungseinheit (MMU) verfügt, die einen Speicherschutz gewährleistet.
Das Fehlen einer MMU bewirkt auch, daß den Nutzerprozessen im Hauptspeicher jeweils ein Bereich von mindestens 64 kByte zugewiesen werden muß, auch wenn das Programm viel kürzer ist.

Da der Kern des MUTOS 1700 einen Adreßraum von 64 kByte belegt, können sich je nach Hauptspeicherausbau drei oder mehr Nutzerprozesse parallel im Hauptspeicher befinden. Weitere aktivierte Prozesse werden über einen Swap-Algorithmus ausgelagert. Entsprechend dem Timesharing-Verfahren werden sie für das Abarbeiten ihrer Zeitscheibe in einen freien Hauptspeicherbereich eingelesen. Eventuell muß dafür ein anderer Prozeß ausgelagert werden.
Aus diesen Darlegungen ist ersichtlich, daß sich die MUTOS 1700-Arbeitsweise nicht grundsätzlich von der des MUTOS 8000 auf dem A 5120.16 unterscheiden wird. In beiden Fällen ist ein Maximalausbau des Externspeicher empfehlenswert.
Der A 7100 ermöglicht es, den Hauptspeicher auf über 256 kByte zu erweitern. Auch das ist für die Arbeit des MUTOS 1700 von Vorteil.
Gegenüber den MUTOS-Systemen auf K 1630, CM4 und A 5120.16 besteht auf dem A 7100 die Möglichkeit, auch Prozesse mit getrennten Befehls- und Datenräumen abzuarbeiten. Derartige Prozesse können 64 kByte Code und 64 kByte Daten umfassen. Interessant ist das besonders für große Programme, die unter den anderen MUTOS-Varianten wie z. B. lint nicht lauffähig sind.

4.
Kompatibilität zu MUTOS 1630

Die bisherigen Ausführungen weisen eigentlich schon darauf hin, daß zwischen MUTOS 1700 und MUTOS 1630 eine sehr hohe Kompatibilitätsstufe erreicht wird. Es werden mit MUTOS 1700 die gleichen Leistungen des Kerns, die gleichen Steuerprogrammrufe, die gleichen Bibliotheksfunktionen, der gleiche Sprachumfang des C-Compilers und der volle Umfang der Systemprogramme angeboten, wie sie von MUTOS 1630 her bekannt sind. Es sind alle Programme lauffähig, die in der Sprache C unter MUTOS 1630 entwickelt wurden, solange sie keine Hardware-Voraussetzungen erfordern, die der A 7100 nicht erfüllt. Einschränkende Bedingungen sind z. B. die begrenzte

Das echtzeitorientierte Betriebssystem BOS 1810

Dr. Peter Kühlborn, Manfred Synowzik
VEB Robotron-Elektronik Dresden

Kapazität von Externspeichern, abweichende Terminaleigenschaften, Zeitprobleme auf Grund höherer Zugriffsdauer zum Externspeicher und das Fehlen spezieller peripherer Geräte (z. B. Magnetband). Eine Erweiterung gegenüber MUTOS 1630 ist die im Abschnitt 3 genannte Möglichkeit, Programme mit getrennten Adreßräumen für Code und Daten abzuarbeiten (magic number 411). Ein weiterer Vorteil des A 7100 ist die höhere Verarbeitungsgeschwindigkeit des Prozessors gegenüber K 1630, die die langsameren Externspeicherzugriffe in vielen Fällen kompensiert. Unter den genannten Randbedingungen kann von einer Aufwärtskompatibilität MUTOS 1630 zu MUTOS 1700 gesprochen werden. Ein Datenaustausch zwischen den Systemen ist über Disketten problemlos möglich. Außerdem besteht über die Kopplungskomponente uucp die Möglichkeit der Online-Kopplung beider Systeme.
Damit ist, wie bei allen MUTOS-kompatiblen Systemen, der Übergang zu einer neuen Architektur mit minimalem Aufwand für den Anwender möglich. Er findet auf dem neuen Rechner die gewohnte Programmierumgebung vor, hat keine größeren Akzeptanz- und Lernprobleme zu lösen und kann Programmpakete mit geringem Anpassungsaufwand weiterhin nutzen – Bedingungen, die für das Betriebssystem MUTOS sprechen.

Literatur

/1/ Anwenderdokumentation MUTOS 1630, VEB Robotron-Projekt Dresden, 1985
/2/ Grützbach, H.: PSU – Eine die Projektierung und Programmierung stützende Umgebung, rechentechnik/datenverarbeitung 20 (1983) 7
/3/ Fischer, G., Clauß, M.: Die Nutzung von PSU im Dialog unter TSO, rechentechnik/datenverarbeitung 22 (1985) 9, S. 23–25
/4/ Bündig, B., Lange, H.-H., Ulrich, P.: MUTOS 8000 – das Betriebssystem des erweiterten Bürocomputers robotron A 5120.16, NTB 29 (1985) 4, S. 149–152
/5/ Bündig, B.: Fröhlich, O.; Lange, H.-H., Ulbrich, P.: Das Betriebssystem MUTOS 8000. edv aspekte Nr. 1, 1986.

1. Allgemeines

Das Betriebssystem BOS1810 ist ein Multitasking-Realtime-System, das für den Einsatz des A 7100 mit seinen spezifischen Anwendungsgebieten und auch besonders für den OEM-Einsatz des MMS 16 vorgesehen ist. Es ist ein modular aufgebautes Betriebssystem mit einer objektorientierten Struktur, das in weiten Grenzen konfigurierbar und somit für die unterschiedlichsten Anwendungen optimal anpaßbar ist.
BOS1810 kann in seiner Doppelfunktion als anwendungsspezifisches Betriebssystem und als Software-Entwicklungssystem genutzt werden. Analog können die Konfigurationen des A 7100 als Ziel- und Wirtsrechner genutzt werden. Für die Programmerstellung und -testung ergeben sich daraus viele Vorteile, weil sowohl die Hardware- als auch die Softwarekomponenten im Ziel- und im Wirtsrechner identisch ausgelegt sein können.
Aus einem einheitlichen Modulpaket kann der Anwender sein eigenes anwendungsspezifisches Betriebssystem mit unterschiedlicher Einsatzcharakteristik und Leistungsfähigkeit konfigurieren. Dabei lassen sich Minimalkonfigurationen, die auf Basis des Exekutivekerns nur 17 KBytes Operativspeicher benötigen, genauso realisieren, wie umfangreiche Konfigurationen mit vollausgebautem Leistungsumfang oder als Softwareentwicklungssystem.
Zur Nutzung des Betriebssystems BOS1810 als Softwareentwicklungssystem werden außer der unmittelbaren Unterstützung des Rechners und seiner möglichen Peripherie auch technologische Mittel zum Erstellen und Austesten der Anwendungssoftware sowie zur Wartung und Pflege von Bibliotheken und Dateien bereitgestellt.

2. Gerätekonfiguration

Das Betriebssystem BOS1810 setzt folgende Minimalkonfiguration voraus:

- Prozessor K 1810 WM86
- programmierbare Interruptsteuereinheit K 580 WN 59A
- 17 KBytes Operativspeicher K 580 WI 53.

Die 17 KBytes Operativspeicher stellen den minimalen Eigenbedarf der Exekutive dar. Dieser Speicher kann sich aus ROM und RAM zusammensetzen. Beim Einsatz von ROM muß beachtet werden, daß die ersten 1024 Bytes zusammenhängender RAM sein müssen, dessen Adreßraum mit der Adresse 0 beginnt. Des weiteren werden noch 2 KBytes RAM als Arbeitsspeicher benötigt.
Die Exekutive ist prinzipiell so ausgelegt, daß sie mit Geräte-Driver für einsatzfallspezifische Peripherie erweitert werden kann. Für eine Reihe von Geräten, die u.a. auch Voraussetzung für den Einsatz des Systems als Softwareentwicklungssystem notwendig sind, enthält BOS1810 bereits die entsprechenden Geräte-Driver. Damit wird auch folgende Standardperipherie des A 7100 bedient:

- zwei Minifolienspeicherlaufwerke (im Grundgefäß des A 7100), K 5600.20 (500 KBytes unformatiert)
- eine Bildschirmeinheit, K 7229
- eine Tastatur, K 7637 (K 76nn)
- ein Seriendrucker mit Centronics-Interface, K 6311.

Zusätzlich können vom BOS1810 bedient werden:

- In einem Beistellgefäß zwei Minifolienspeicherlaufwerke K 5600.20 (500 KBytes) oder zwei Standard-Folienspeicherlaufweke K 602.10 (800 KBytes/vorerst 400 KBytes unformatiert)
- ein ASP mit den Interfaces ... 1 IFSS ... 1 IFSP ... 1 V.24
- alternativ zur Bildschirmeinheit und Tastatur eine Bedieneinheit K 8911.

3. Leistungen des Betriebssystems BOS1810

3.1. Exekutive des BOS1810

Die Exekutive besteht aus mehreren wahlweise nutzbaren Subsystemen. Innerhalb der Subsysteme stehen wiederum wahlweise Leistungen bereit. Der Nutzer kann aus beiden die für seinen Einsatzfall notwendigen Leistungen auswählen und die Exekutive so konfi-

gurieren, daß sie dem Einsatzfall optimal angepaßt ist.
Das BOS1810 wird durch folgende Eigenschaften charakterisiert:
– Simultanes Überwachen und Steuern von unabhängigen externen Ereignissen
– Möglichkeit des Bedienens einer Vielfalt von E/A-Geräten und externen Speichern
– Bereitstellen wirkungsvoller und flexibler Mittel für den Operator, um das Systemverhalten zu beobachten und zu verändern
– Bereitstellen von Übersetzern und weiterer Hilfsmittel zur Programmentwicklung.
Die Systemleistungen des BOS1810 lassen sich wie folgt gliedern:
– Systemarchitektur
– Ein-/Ausgabeorganisation
– Nutzeranpassung
– Hilfsmittel.

3.1.1. Systemarchitektur

BOS1810 ist ein objektorientiertes Betriebssystem. Es stellt eine Reihe von Bausteinen zur Verfügung, mit deren Hilfe der Anwender seine Anwendungssoftware aufbaut. Diese Bausteine werden Objekte genannt. Solch eine objektorientierte Struktur macht ein Betriebssystem überschaubar sowie anwenderfreundlich und damit sicherer in der Anwendung.
Das Betriebssystem unterscheidet verschiedene Typen von Objekten: Tasks, Jobs, Mailboxen, Semaphore, Segmente, Connections und andere, die vom Anwender definiert bzw. erzeugt werden können und stellt Systemrufe zur Verfügung, mit denen der Nutzer in seinem Programm die Objekte manipulieren kann (z. B. CRATE MAILBOX und DELETE MAILBOX).
Jeder Objekttyp verfügt über spezielle Eigenschaften, die über entsprechende Systemrufe ansprechbar sind. Jedem Objekttyp ist ein Satz von Systemrufen zugeordnet, d. h., der Nutzer braucht nur die Objekte, deren Eigenschaften und die dazugehörigen Systemrufe zu kennen, die er in seinem Programm benötigt.

Die Exekutive unterstützt Multitasking, um die Entwicklung der Anwendungssoftware, die Echtzeitereignisse verarbeitet, zu vereinfachen. Ein einziges Programm, das versucht, mehrere Ereignisse zu verarbeiten, wird zwangsläufig sehr komplex. Die Komplexität eines solchen Programms vergrößert sich mit der Anzahl der zu überwachenden Ereignisse. Besser und übersichtlicher ist es, mehrere Programme zu schreiben, wobei jedes Programm ein Ereignis bearbeitet. Jedes dieser Einzelprogramme bildet eine Task, einen Objekttyp, der von der Exekutive unterstützt wird. Tasks sind die einzigen aktiven Objekte, nur sie können demzufolge Systemrufe aussenden.
Multitasking und Interrupt-Behandlung stehen in enger Beziehung. Wenn ein Interrupt auftritt, wird eine Task aktiviert, um den Interrupt zu bearbeiten. Wegen der direkten Beziehung zwischen Interrupts und Tasks ist eine Anwendungslösung, die auf BOS1810 basiert, einfach erweiterbar. Für jede neue Interruptquelle muß lediglich ein weiteres Objekt vom Typ Task in das System integriert werden, das das Bearbeiten des neuen Interrupts übernimmt.
Die Leistungen der Multitasking und Interrupt-Bearbeitung können erst mit der Unterstützung einer prioritätsgesteuerten Ablaufplanung voll ausgeschöpft werden. Bei der prioritätsgesteuerten Ablaufplanung wird jeweils die abarbeitungsfähige Task, die die höchste Priorität besitzt, abgearbeitet. Wenn also eine laufende Task oder ein Interrupt eine Task höherer Priorität aktiviert, wird die laufende Task unterbrochen und zur Task mit der höheren Priorität verzweigt, falls diese abarbeitungsfähig ist. Nach Beendigung der Task mit der höheren Priorität wird die unterbrochene Task weiterbearbeitet.
BOS1810 unterstützt eine Priorität zwischen 0 und 250, wobei die Priorität von 250 nach 0 steigt. Die Prioritäten der Tasks können entsprechend der Bedeutung der Ereignisse, die sie bearbeiten, gewählt werden.
Die Muliprogrammierung ist eine Technik, die das gleichzeitige Bearbeiten mehrerer Anwendungen auf einem Rechner ermöglicht. Damit sich die Anwendungen nicht gegenseitig beeinflussen, muß jede mit einer eigenen Umgebung ausgestattet werden. Das bedeutet, jede Anwendung erhält ihren eigenen Speicher, ihre eigenen Dateien und ihre eigenen Objekte. BOS1810 unterstützt einen Objekttyp, den Job, der solch eine Abgrenzung von einzelnen Anwendungen erleichtert. Ein Job ist ein passives Objekt. Jeder Job enthält in der Regel eine Anzahl von Tasks und alle Ressourcen, die diese Tasks benötigen. Jobs stellen günstige Grenzen für die dynamische Speicherzuweisung dar und sind im Fehlerfalle eine günstige Einheit zum Löschen.
Neben der Abgrenzung einzelner Anwendungen untereinander spielen bei den Echtzeitsystemen die Möglichkeiten des gegenseitigen Beeinflussens von Tasks eine wichtige Rolle. Deshalb verfügt BOS1810 über wirkungsvolle Mittel zur Task-Koordinierung. Tasks können sich auf verschiedene Weise beeinflussen. Sie können Informationen austauschen, sich gegenseitig ausschließen oder sich synchronisieren. Die Mittel, über die diese Leistungen realisiert werden können, sind spezielle Objekttypen. Der Informationsaustausch erfolgt über den Objekttyp Mailbox, wobei sich an jeder Mailbox für Sender- und Empfängertasks Warteschlangen aufbauen können. Der gegenseitige Ausschluß von Tasks kann über die Objekttypen Semaphor und Region realisiert werden. Ein Semaphor ist ein Integer-Zähler, dem über entsprechende Systemrufe Einheiten gesendet oder abgezogen werden. In Abhängigkeit vom Zählerstand (Wert des Semaphors) und der Anzahl der abgeforderten Einheiten kann die abfordernde Task zwei Zustände einnehmen. Ist der Inhalt des Zählers größer oder gleich dem abgeforderten Wert, wird die Task weiterbearbeitet. Im anderen Falle wird die Task in einen Wartezustand versetzt. Eine Region ist ein Datenbereich, zu dem zu einem Zeitpunkt jeweils nur eine Task Zugriff haben kann. Die Synchronisation kann analog zum gegenseitigen Ausschluß über den Objekttyp Semaphor realisiert werden.
Die objektorientierte Struktur des

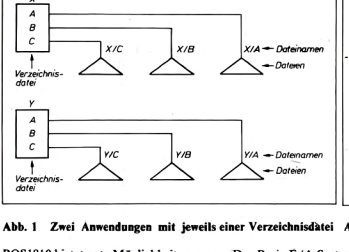


Abb. 1 Zwei Anwendungen mit jeweils einer Verzeichnisdatei

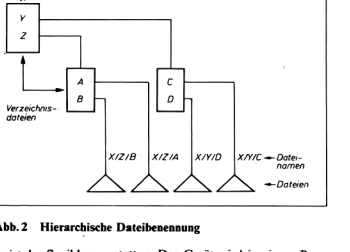


Abb. 2 Hierarchische Dateibenennung

BOS1810 bietet gute Möglichkeiten zum Erweitern des Systems. Der Nutzer kann sich eigene Objekte und die notwendigen Systemrufe zu ihrer Manipulation schaffen, die nach ihrem Einbinden in das System genau wie die systemeigenen Objekttypen behandelt werden.

Das Austesten der Anwendungssoftware wird bei BOS 1810 mit einer objektorientierten Testunterstützung erleichtert. Zur Verfügung stehen der Dynamische Debugger, kurz Debugger und der System-Debugger. Da diese Testhilfen die Objekte der Exekutive mit berücksichtigen, können beim Programmtest auch die Beziehungen der Tasks untereinander überprüft und überwacht werden. Der Debugger ermöglicht das Testen von einzelnen Tasks, während das System weiterarbeitet. Der System-Debugger ermöglicht das Prüfen der Speicher- und Registerinhalte sowie eine Statusüberprüfung aller zur Zeit genutzten Objekte bei gestopptem System.

3.1.2. Ein-/Ausgabeorganisation

Um die differenzierten Anforderungen vieler Anwendungen an die E/A-Leistungen zu befriedigen, verfügt BOS1810 über zwei E/A-Systeme. Es können das Basis-E/A-System oder eine Kombination aus Basis-E/A-System und erweitertem E/A-System eingesetzt werden.

Das Basis-E/A-System ist das flexiblere der beiden E/A-Systeme. Es bietet E/A-Leistungen, die für zeit- und speicherkritische Anwendungen sinnvoll sind. Sie erlauben es, das System hinsichtlich seiner Leistungsfähigkeit zu optimieren. Es setzt allerdings beim Anwender größere Kenntnisse über die E/A-Organisation voraus. So obliegt es dem Anwender, die E/A-Operationen zu synchronisieren und falls benötigt, eine eigene Pufferorganisation für die Datenübertragungen zu schaffen. Das Basis-E/A-System ist für Direktzugriffsoperationen besser geeignet als das erweiterte E/A-System.

Das Erweiterte-E/A-System ist für den Nutzer leichter zu handhaben und nimmt ihm lästige Detailaufgaben und Parameterangaben ab. Es beschleunigt mit Hilfe einer entsprechenden Pufferorganisation das Verarbeiten der Ein- und Ausgabedaten (reading ahead/writing behind). Das Erweitere-E/A-System ist auf Grund seiner vorausschauenden Arbeitsweise bei den sequentiellen E/A-Operationen besonders wirkungsvoll.

Mit einem einheitlichen Satz von Systemrufen zur Kommunikation mit den E/A-Geräten unterstützt BOS1810 geräteunabhängige E/A-Operationen. So gibt es für Ein- und Ausgaben die Systemrufe READ und WRITE, die eine leichte Handhabung der Ein- und Ausgaben durch die Nutzerprogramme gestatten. Das Gerät wird in einem Parameter des Rufes spezifiziert. Mit einer Variablen als Geräteparameter können völlig geräteunabhängige E/A-Prozeduren geschaffen werden.

Die Möglichkeit der hierarchischen Benennung von Dateien auf Massenspeichern vereinfacht den Prozeß der Organisation der Dateibenennung und erhöht die Flexibilität des Systems. Die praktische Anwendung sieht so aus, daß z. B. bei der Multiprogrammierung jeder Anwendung eine spezielle Datei zugeordnet wird, die ein Verzeichnis aller zu dieser Anwendung gehörigen Dateien enthält. Das Führen der Dateinamen in solch einer Verzeichnisdatei gestattet die Nutzung gleicher Dateinamen in verschiedenen Anwendungen eines Anwendersystems (Abb. 1). Mehrere Verzeichnisdateien können wiederum in einer übergeordneten Verzeichnisdatei zusammengefaßt werden. Dieser Vorgang kann sich nun beliebig oft wiederholen. Die Bezugnahme auf eine Datei erfolgt so, daß immer zuerst die Verzeichnisdatei und dann die gewünschte Datei angegeben werden muß. Bei mehreren hierarchisch angeordneten Verzeichnisdateien ist dann immer der ganze Pfad, beginnend mit der höchsten Verzeichnisdatei und endend mit dem Namen der gewünschten Datei, anzugeben (Abb. 2). Dieser Art der Dateibennung erlaubt das Erweitern des Dateibestandes um eine neue Anwendung. In die-

sem Fall erhält die neue Anwendung eine eigene Verzeichnisdatei, die die dazugehörigen Dateinamen enthält. Die Dateinamen können somit ohne Berücksichtigung der bereits in anderen Anwendungen vorhandenen Dateinamen gewählt werden.
Die Dateizugriffssteuerung des BOS1810 erlaubt dem Anwendersystem, den Zugriff zu hierarchisch benannten Dateien lenkend zu beeinflussen. Der Eigentümer einer Datei kann angeben, wer eine Datei nutzen darf und wie sie genutzt werden soll. Praktisch kann jeder Dateieigentümer jedem weiteren Nutzer sogar bestimmte Kombinationen des Zugriffs (nur lesen, nur schreiben, lesen und schreiben usw.) gewähren. Auf diese Weise kann sich jeder Nutzer einen leistungsfähigen Dateischutz realisieren.
Mit der Steuerung der Dateiaufteilung, d.h. Festlegen der Blocklänge jeder Datei auf dem Massenspeicher, kann der Nutzer die Zugriffszeiten zu den Dateien und die Ausnutzung des Speicherplatzes auf dem Massenspeicher beeinflussen. Soll das Zeitverhalten günstig beeinflußt werden, empfiehlt es sich, mit relativ großen Blöcken zu arbeiten. Eine Vergrößerung der Blocklänge führt in der Regel zu einer Steigerung der Übertragungsrate sowie zu einer Reduzierung der Zugriffszeiten. Andererseits führt aber eine größere Blocklänge zu einer ungünstigen Ausnutzung des Speicherplatzes. In den meisten Fällen wird bezüglich der Blocklänge ein sinnvoller Kompromiß geschlossen werden müssen.
Die Flexibilität eines Systems wird auch von der Auswahl der Geräte-Driver geprägt. Zum Liefersatz des BOS1810 gehören die Geräte-Driver für die Standard-Peripherie. Der Nutzer kann diese Palette noch um eigene Driver für anwendungsspezifsiche Geräte erweitern. In das System werden dann nur die Driver integriert, die zur Lösung der speziellen Anwendung benötigt werden.
Der Terminal-Support-Code des BOS1810 ist ein Programmpaket, das als programmierbares Interface zwischen einem Terminal-Driver und dem E/A-System dient. Damit wird eine gewisse Terminal-Unabhängigkeit geschaffen. Der Nutzer kann unterschiedliche Terminal-Typen (mit KO17-Tastatur) bedienen, ohne seine Anwendungssoftware ändern zu müssen. Weiterhin ist es möglich, Sonderbehandlungen (für spezielle Zeichen und Steuerzeichen) festzulegen. Diese Terminal Charakteristika können über das Programm oder vom Bediener festgelegt werden. Dabei können sie in Systemen mit mehreren Terminals für jedes Terminal unterschiedlich sein.

3.1.3. Nutzeranpassung

Da die Anforderungen an ein Echtzeitsystem, speziell auch bei OEM-Einsätzen, von Einsatzfall zu Einsatzfall sehr unterschiedlich sein können, sind verschiedene Teile des BOS1810 bezüglich ihrer Leistungen an die Gegebenheiten der einzelnen Einsätze anpaßbar.
So unterstützt BOS1810 die Modifikation vorhandener bzw. die Entwicklung neuer interaktiver Nutzerkommandos. Mit dem Schaffen von Kommandos, die auch speziell die Qualifikation des Bedienpersonals berücksichtigen, kann das Kommunikationsniveau zwischen Mensch und Maschine beeinflußt werden. Auf diese Weise kann ein Anwendungssystem bedienerfreundlich gestaltet werden, so daß die Möglichkeiten für das Bedienpersonal, Fehler zu machen, reduziert werden.
Die neuen Kommandos werden als Programmdaten auf einem Massenspeicher stationiert und bei Bedarf unter Steuerung des Human-Interface (zentrales Kommandoprogramm) in den dynamisch verwalteten Teil des Systemspeichers geladen und abgearbeitet.
Zum Laden der Anwendungssoftware verfügt BOS1810 über ein leistungsfähiges Ladeprogramm. Anwenderprogramme können so vom Massenspeicher eingelesen und abgearbeitet werden. Weiterhin besteht die Möglichkeit, Programme in Segmente zu zergliedern und diese als Überlagerungsstrukturen zu behandeln, so daß nicht das gesamte Programm gleichzeitig im Hauptspeicher stehen muß. Das Ladeprogramm gibt dem Nutzer viel Bewegungsfreiheit hinsichtlich der Nutzung des Hauptspeichers durch die Programme. Die Programme können jeweils dorthin geladen werden, wo sich gerade freier Speicherbereich befindet.
BOS1810 ist ein Mehr-Terminal-System, das zwei Möglichkeiten der simultanen Unterstützung mehrerer Terminals bietet. Die erste Möglichkeit ergibt sich aus dem Hüman-Interface für Mehrnutzerbetrieb. In diesem Falle handelt es sich um eine Kommunikation auf höherer Ebene. Es können von mehreren Terminals Kommandos eingegeben sowie Dienstprogramme, Übersetzer und Anwendungsprogramme gestartet und bedient werden. Beim Zuschalten eines Terminals weist die Exekutive dem Terminal eine Arbeitsumgebung zu. Diese Arbeitsumgebung besteht aus einem Identifikationscode (ID), einem Speicherbereich, in dem Programme abgearbeitet werden können sowie einer Priorität für diese Programme. Außerdem erfolgt der Start eines Anfangsprogramms, das im Standardfall ein Kommandozeileninterpreter ist, der vom System bereitgestellt wird. Der Nutzer hat aber auch die Möglichkeit, ein eigenes Anfangsprogramm einzusetzen.
Die zweite Möglichkeit besteht darin, daß der Nutzer eigene Software für diese Zwecke einsetzt. Der Verkehr mit den Terminals wird aus den Nutzerprogrammen heraus über die E/A-Systemrufe realisiert. Dieses zweite Verfahren ist für die Fälle gedacht, wo Funktionen benötigt werden, die die vorhandene Mehr-Terminal-Unterstützung nicht bieten bzw. wo ohne Human-Interface gearbeitet werden soll.
Mit dem Binden zur Laufzeit bietet BOS1810 eine Möglichkeit der dynamischen Modifikation des Systems. Realisiert werden drei Arten der Laufzeitbindung: Binden von Objekten an Tasks, Binden von Dateien und Geräten an Tasks und Binden der Anwendungssoftware an das BOS1810. Dieses Binden zur Laufzeit macht ein Anwendungssystem sehr beweglich. Mit der Möglichkeit des Benennens von Objekten bietet BOS 1810 ein Mittel zur gemeinsamen Nutzung von dynamisch erzeugten Ob-

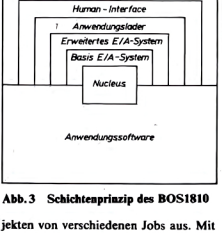


Abb. 3 Schichtenprinzip des BOS1810

jekten von verschiedenen Jobs aus. Mit der Einführung logischer Namen für Dateien und Geräte ist es möglich, den Zugriff zu Dateien und Geräten dynamisch zu gestalten. Die Tasks müssen nicht für eine feste Geräte- bzw. Dateikombination ausgelegt werden. Mit Hilfe des Software-Interrupts (Systemrufe) zum Binden der Anwendungssoftware an die Exekutive des BOS1810 kann diese neu konfiguriert werden, ohne daß die Anwendungssoftware neu übersetzt bzw. der Link-Vorgang wiederholt werden muß.
Die Fehlerbehandfung des BOS1810 umfaßt den Prozeß des Erkennens und der Reaktion auf unerwartete Bedingungen. Die Exekutive unterstützt diesen Prozeß mit einer beträchtlichen Anzahl von Gültigkeitstests und Bedingungskontrollen. Gehen diese Tests und Kontrollen negativ aus, gibt es grundsätzlich zwei Möglichkeiten der Reaktion. Eine davon, das automatische Aufrufen einer zentralen Fehlerbehandlungsroutine von der Exekutive, vereinfacht die Fehlerbehandlung entscheidend. Diese Fehlerbehandlungsroutine kann sich der Nutzer selbst schreiben, er kann aber auch die Standard-Fehlerbehandlungsroutine des BOS1810 nutzen. Die zweite Methode, bei der sich die Nutzertask selbst mit den Ausnahmebedingungen befaßt, ermöglicht ein spezielles Behandeln von Ausnahmesituationen. Mögliche Reaktionen auf solche Ausnahmesituationen sind: Der Fehler wird behoben, die Task, die den Fehler verursacht, wird gelöscht, der Bediener wird gewarnt oder der Fehler wird generell übergangen.
BOS1810 unterstützt eine effektive Speicherausnutzung mit einer dynamischen Speicherzuweisung. Wenn auf einem Anwendersystem mehr als eine Anwendung läuft, können sich die Speicheranforderungen der verschiedenen Jobs ergänzen. Das bedeutet, ein Job gibt einen Speicherbereich frei und ein anderer fordert diesen an. Gleiches gilt natürlich auch für Tasks innerhalb eines Jobs. Auf diese Weise wird ein Teil des verfügbaren Hauptspeichers mehrfach genutzt. Die dynamische Hauptspeicherzuweisung kann so den für ein Anwendersystem notwendigen Speicherbedarf erheblich reduzieren.

3.1.4. Hilfsmittel

Zusammen mit der Exekutive des BOS1810 wird ein Satz von Software-Hilfsmitteln zur Programmentwicklung mitgeliefert. Es gibt nun Einsatzfälle, die diese Hilfsmitteln ständig benötigen, und solche, die nur bei der Entwicklung der Anwendungssoftware vorhanden sind. Der Vollständigkeit halber soll hier noch einmal auf die bereits erwähnten Debugger eingegangen werden.
Der Debugger ist speziell an die objektorientierte Struktur des Systems angepaßt. Er besitzt zwei Eigenschaften, die den Prozeß der Fehlersuche in einer Multitasking-Umgebung stark vereinfachen. So erlaubt er das Testen einer oder mehrerer Tasks, während das System in Echtzeit weiterarbeitet. Außerdem kann der Debugger anzeigen, welche Tasks und Objekte in Warteschlangen an Mailboxen und Semaphoren stehen. Da die Arbeit mit diesem Debugger das Echtzeitverhalten des Systems nicht beeinflußt, können während des laufenden Betriebes neue Programme ausgetestet werden.
Zu BOS1810 gehört außerdem noch ein System-Debugger, der die Möglichkeiten des Monitorprogramms (eine Testunterstützung, die mit dem Rechner auf ROM geliefert wird) erweitert und der ebenfalls an die objektorientierte Struktur des BOS1810 angepaßt ist. Er bietet Leistungen, die u. a. zur Diagnose von Systemabbrüchen notwendig sind. Der System-Debugger erleichtert das Rückverfolgen von Feldern bei gestoppten System, indem er Informationen über die Systemrufe, die Objekte und den Stack einer Task bereitstellt. Zum Liefersatz des BOS1810 gehört ein Basissystem, ein arbeitsfähiges System, das als Software-Entwicklungssystem und als Zielsystem genutzt werden kann. Wird ein spezielles Zielsystem für eine Anwendung benötigt, kann sich der Nutzer aus den Dateien des Liefersatzes, die die Bausteine des Betriebssystems enthalten, mit Hilfe des Basissystems dieses neue Zielsystem konfigurieren. Er kann aber auch mittels Modifikation der Konfigurierungsdatei, die zum Erzeugen des Basissystems erstellt wurde, anwendungsspezifische Zielsysteme herstellen. Das Basissystem ist außerdem ein Hilfsmittel, um sich schnell praktische Fertigkeiten im Umgang mit dem System zu erwerben.
Wie schon erwähnt, ist das BOS1810 ein Bausteinsystem, aus dessen Liefersatz sich der Nutzer (mit Hilfe einer interaktiven Konfigurierung) sein Zielsystem herstellen kann. Der Vorgang der Konfigurierung ist einfach und sicher, weil ein entsprechendes Dienstprogramm den Ausführenden durch den Prozeß führt. Als Ergebnis des Konfigurationsvorganges entsteht eine Konfigurations-Datei, mit der das Erzeugen eines Systems mehrfach wiederholt werden kann. Diese Konfigurierungs-Datei kann aber auch nach der Modifikation als Ausgang für modifizierte Systeme genommen werden. Die Exekutive besteht aus einer Anzahl von Subsystemen, die z. T. hierarchisch aufeinander aufbauen (Abb. 3). Jede Schicht des BOS1810 benötigt jeweils alle darunterliegenden Schichten. Die Anwendungssoftware kann alle Schichten auch direkt nutzen. Die einzelnen Subsysteme enhalten z. T. Standard-Leistungen und wahlfreie Leistungen. Bei der Konfigurierung erfolgt die Auswahl der für die Anwendung benötigten Subsysteme und wahlfreien Leistungen. Der Nucleus ist das Herz der Exekutive des BOS1810.

Alle anderen Teile der Exekutive bauen auf dem Nucleus auf, so daß jedes Anwendersystem, das auf BOS1810 basiert, den Nucleus enthalten muß. Die weiteren wahlfreien Subsysteme sind: das Basis-E/A-System, das Erweiterte-E/A-System, das Human-Interface, der Anwendungslader, der Debugger, der System-Debugger, der Terminal-Handler und das Universelle Entwicklungsinterface (eine standardisierte Schnittstelle zwischen Anwendungssoftware und Exekutive, die neben den eigentlichen Systemrufen existiert).
Eine Anzahl Dateiwartungsprogramme, die als Human-Interface-Kommandos bereitgestellt werden, runden die Palette der Hilfsmittel ab. Da die Kommandonamen mnemonisch sind, soll hier folgende Auswahl nur aufgezählt werden: COPY, FORMAT, DIR, RENAME, CREATEDIR, SUBMIT und BAKKUP/RESTORE.

3.2.
Dienstprogramme

Neben den der Exekutive zugeordneten Hilfsmitteln zur Programmentwicklung gehören zum BOS1810 noch einige Dienstprogramme. Da die meisten Dienstprogramme hinsichtlich ihrer Leistungen von System zu System ähnlich ausgelegt sind, sollen sie hier nur kurz erwähnt werden:
– Linker
Der Linker verbindet Objektmoduln, die getrennt und von verschiedenen Übersetzern übersetzt sein können, zu einem verschieblichen Objektmodul.
– Locator
Der Locator teilt den verschieblichen Objektmoduln des Linkers absolute Adressen zu.
– Bibliothekar
Mit dem Bibliothekar werden Objektmodulbibliotheken erzeugt und verwaltet sowie Inhaltsverzeichnisse und externe Symbole ausgelistet.
– Zeileneditor und bildschirmorientierter Editor
Die Editoren dienen dem Erzeugen und Aufbereiten von Quellprogrammen und Textdateien.
– Cross-Referenz-Programm
Es verarbeitet verschiedene Objektmoduln und liefert eine Druckliste der im Modul definierten und verwendeten globalen Symbole.
– Hexadezimalkonvertierungsprogramm
Das Konvertierungsprogramm wandelt Moduln mit absolutem Objektformat in solche mit hexadezimalem Format um.
– Arithmetik-Objektbibliotheken
Sie liefern die Voraussetzung zur Nutzung von Grundfunktionen und Arithmetik. Die Bibliotheken werden vom Linker mit dem Anwendungssystem verbunden.

3.3.
Sprachübersetzer und/oder -interpreter

Der Assembler ASM 86 übersetzt Quellprogramme, die in der Assemblersprache ASM86 geschrieben sind, in zwei Durchläufen in einen verschieblichen Objektcode, der vom Linker weiterverarbeitet werden kann. Zusätzlich kann wahlweise die Ausgabe einer Assembler- und/oder Fehlerliste sowie einer Cross-Referenz-Tabelle angewiesen werden.
Außerdem können in das zu übersetzende Quellprogramm externe Quelldateien (INCLUDE-Dateien) einbezogen werden.
Die Makrotechnik, die unter anderem eine bedingte Assemblierung ermöglicht, gestattet dem Anwender, eigene Makros zu definieren und zu verwenden.
Die Programmiersprache PLM 86 ist eine höhere Sprache sowohl für die System- als auch für die Anwendungsprogrammierung im wissenschaftlich-technischen und im ökonomischen Bereich sowie für die Programmierung von Aufgaben der Prozeßsteuerung und -verarbeitung.
PLM86 unterstützt das strukturierte Programmieren mittels Blockkonzept und Programmablauf-Strukturen.
Von PLM86-Programmen aus können sämtliche Prozeduren der Systemkomponenten des BOS1810 direkt gerufen werden. Der PLM86-Compiler übersetzt Quellprogramme, die in dieser Programmiersprache geschrieben sind, in verschieblichen Objektcode und unterstützt das Erstellen und Austesten von PLM86-Programmen. Gesteuert werden können sowohl das Erzeugen des Objektcodes hinsichtlich Optimierung, Speicherverwaltung, Laufzeit-Fehlermeldungen u. ä. als auch das Erzeugen der Übersetzungsliste hinsichtlich Inhalt und Form.
Die Programmiersprache FORTRAN-77 ist die aktuelle standardisierte Sprachversion von FORTRAN (ANSI X3.9 – 1978). Der FORTRAN-Compiler übersetzt ein FORTRAN-77 Quellprogramm in die Assemblerprogrammiersprache. Nach dessen Assemblierung und Verbindung mit dem FORTRAN-Laufzeitsystem (mittels LINKER) entsteht ein ausführbares Programm. Der Compiler arbeitet als Mehrpaß-Compiler und ist über die Auswahlbedingungen steuerbar. Neben der Quelltextausgabe ist die Ausgabe einer Attribut- und Cross-Referenz und die Optimierung des Zielcodes möglich.

Lieferbar

edv-aspekte 4/1986
Unter dem Titel
Grundlagen und Anwendung der Informatik

bringt diese edv-aspekte-Ausgabe Beiträge vom vierten Symposium der Sektion Informatik der Technischen Universität Karl-Marx-Stadt zum gleichnamigen Thema.
Die Beiträge wurden nach drei Themen geordnet:

- CAD/CAM-Anwendungen und Mikrorechnereinsatz
- Methoden und Werkzeuge der Projektierung
- Rationalisierungssoftware für ESER.

Bestellungen sind zu richten an den Verlag Die Wirtschaft, Abteilung Vertrieb, Am Friedrichshain 22, Berlin, 1055.

Hinweis

Wir bitten unsere Leser um Verständnis dafür, daß das System AIDOS/16 aus Platzgründen nicht in diesem Heft vorgestellt werden kann. Der Beitrag wird in der rd 3/87 veröffentlicht.

Sprachkonzeption des A 7100

Dr. Rolf Dottermusch
VEB Robotron-Projekt Dresden

1. Langfristige Zielstellung für die Implementation von Programmiersprachen

In den vergangenen Jahren hat sich international die Erkenntnis durchgesetzt, daß das Anwenden höherer Programmiersprachen eine wesentliche und ganz entscheidende Voraussetzung für die Rationalisierung und Effektivierung der Softwareentwicklung ist. Sie beeinflussen in entscheidendem Maße das Verkürzen der Entwicklungszeiten, senken den Wartungsaufwand und bieten überhaupt erst die Gewähr für die Portierbarkeit von Software. Obwohl in allen kommerziell genutzten Betriebssystemen noch Assembler bzw. Makroassembler existieren, werden sowohl Anwendersoftware als auch Systemsoftware in wachsendem Maße in höheren Programmiersprachen implementiert. Dieser Entwicklungstrend wurde in den letzten Jahren auch von der Entwicklung und breiten Nutzung von Systemprogrammiersprachen, durch die Schaffung von leistungsfähigen und komfortablen Programmierungsumgebungen bzw. Programmiersystemen für höhere Programmiersprachen und nicht zuletzt durch die sprunghafte und vielfältige Anwendung der Mikrorechentechnik beeinflußt. Diesem kurz umrissenen internationalen Trend folgend ergibt sich langfristig eine Zielstellung des VEB Robotron-Projekt Dresden. Sie verfolgt folgende Absichten:

- Die Anwender immer besser mit einem kontinuierlichen Angebot an international eingeführten und verbreiteten höheren Programmiersprachen zu befriedigen
- die Portierbarkeit der Anwendersoftware mittels kompatiblen und auf internationale Programmiersprachenstandards orientierte Implementationen zu unterstützen und
- die Anwender auf moderne technologische Prinzipien der Softwareentwicklung durch das Bereitstellen moderner höherer Programmiersprachen und leistungsfähiger Programmierumgebungen zu orientieren.

Die erste Etappe bei der Realisierung dieser Zielstellung ist die Unterstützung eines Kerns des breiten Spektrums an höheren Programmiersprachen auf den Rechner- und Betriebssystemen des VEB Kombinats Robotron, die einen hohen Verbreitungsgrad haben bzw. erwarten lassen. Dieser Kern umfaßt die traditionellen Sprachen FORTRAN (in den standardisierten Versionen FORTRAN IV und FORTRAN 77), COBOL und PASCAL sowie C und MODULA-2. C ist eine Systemprogrammiersprache, die sich gegenwärtig stark verbreitet. MODULA-2 ist eine Programmiersprache mit modernen Konzepten für den Entwurf und die Implementierung von Anwender- und Systemsoftware.

Dieser Kern wird unmittelbar tangiert von PL/1 für Rechner und Betriebssysteme des ESER und von BASIC für die meisten Betriebssysteme der Klein- und Mikrorechner (Abb. 1).

In einer weiteren Etappe sollen die Übersetzerprogramme für die genannten Sprachen in leistungsfähige Programmierumgebungen eingebettet werden. Da für die in den ESER-Betriebssystemen unterstützten Sprachen teilweise Programmierumgebungen und -hilfsmittel existieren (z. B. Testcompiler, Optimierungscompiler und TESYS), konzentrieren sich diese Arbeiten auf die Klein- und insbesondere auf die Mikrorechentechnik.

2. Spektrum der in Betriebssystemen des A 7100 unterstützten Programmiersprachen

2.1. Übersicht

Ausgehend von den genannten Zielstellungen und den Anforderungen des Entwicklers der Betriebssysteme für den A 7100 werden hinsichtlich der Unterstützung höherer Programmiersprachen zunächst die Betriebssysteme SCP 1700 und BOS 1810 berücksichtigt.

Im SCP 1700 erfolgen gegenwärtig Implementierungen für

- BASIC 1700
- FORTRAN 77
- COBOL 1700
- PASCAL
- MODULA-2 und
- C.

Für das BOS 1810 werden Compiler für die Sprachen

- PLM 86 und
- FORTRAN 77

bereitgestellt (Abb. 2).

Wie diese Aufzählung erkennen läßt, wird dem SCP 1700 der Vorrang gegeben. Das ergibt sich aus dem zu erwartenden Verbreitungsgrad und den Anwendungsgebieten dieser Betriebssysteme, auf die hier nicht näher eingegangen werden soll. Außerdem soll darauf hingewiesen werden, daß neben

Abb. 1 Kern des Spektrums von Programmiersprachen

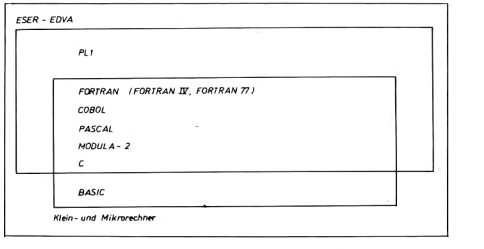


Abb. 2 Sprachspektrum für A 7100

Sprache	SCP 1700	BOS 1810	(MUTOS)
FORTRAN 77	x	x	0
BASIC	BASIC 1700		Standard-BASIC
COBOL	COBOL 1700		0
MODULA-2	x		0
C	x		0
PLM86		x	
PASCAL	x		0

Erklärung: x – Implementierung erfolgt
0 – Portierung beabsichtigt
name – Implementierung der Sprache name

Sprachspektrum für A 7100

PASCAL die Sprache MODULA-2 unterstützt wird. Der PASCAL-Programmierer erhält damit ein effektiveres und leistungsfähigeres Werkzeug. Auf die Zusammenhänge zwischen PASCAL und MODULA-2 wird später eingegangen. Für ein weiteres Betriebssystem des A 7100 (MUTOS) sind Portierungen der im SCP 1700 unterstützten Sprachen beabsichtigt. Eine Ausnahme ist BASIC 1700. Diese Sprachversion von BASIC ist kompatibel zu dem international verbreiteten MBASIC-86. Damit soll dem BASIC-Anwender die Portierung seiner BASIC-Software vom SCP 1520 in das SCP 1700 erleichtert werden. Zukünftig wird aber ein Standard-BASIC entsprechend dem Standardvorschlag von 1984 bereitgestellt, da die zahlreichen unterschiedlichen BASIC-Sprachversionen anders nicht mehr zu berücksichtigen sind.

2.2. Kurzcharakteristik der unterstützten Programmiersprachen

Im folgenden soll auf die einzelnen Sprachen etwas näher eingegangen werden. Dabei ist nicht beabsichtigt, Details zu erläutern. Zur Charakterisierung der einzelnen Sprachen werden einige allgemeine Bemerkungen gemacht auf in der Sprache unterstützten Datentypen, Operationen und Anweisungen. Besonderheiten der Sprache bzw. Unterschiede zu Vorgängersprachen oder entsprechenden Sprachstandards werden hervorgehoben und die Leistungsfähigkeit der Sprachübersetzer kurz erläutert.

FORTRAN 77

Die Sprache FORTRAN 77 ist die Weiterentwicklung der im ANSI 3.9-1966 standardisierten Sprache FORTRAN IV. Die Standardisierung von FORTRAN 77 erfolgte 1978 und ist im ANSI 3.9-1978 beschrieben. In FORTRAN 77 werden die Basisdatentypen INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL und CHARACTER (Zeichen und Zeichenketten)

sowie Felder dieser Datentypen unterstützt. Zur Manipulation der Daten stehen arithmetische Operatoren, Relationsoperatoren, logische Operatoren und der Zeichenkettenoperator Verkettung (//) zur Verfügung. Neben einer Reihe nichtausführbarer Anweisungen (Deklarations- und Spezifikationsanweisungen) gibt es folgende ausführbare Anweisungen:

– Arithmetische, logische, Zeichenketten und Markenzuweisung
– unbedingtes, berechnetes GOTO sowie GOTO mit Markenzuweisung
– die arithmetische IF-Anweisung der Form
IF (arithmetischer ausdruck) marke1, marke2, marke3
– die logische IF-Anweisung der Form
IF (logischer ausdruck) anweisung
– die Block-IF-Anweisung der Form
IF (logischer ausdruck)
THEN anweisungsfolge
ELSE IF (logischer Ausdruck) THEN anweisungsfolge
⋮
ELSE IF (logischer ausdruck) THEN anweisungsfolge
ELSE anweisungsfolge
END IF
(die ELSE-IF-Teile und der ELSE-Teil sind dabei wahlweise)
– die DO-Anweisung
– die CONTINUE-, STOP-, PAUSE- und END-Anweisungen
– die Anweisungen zur Unterprogrammverzweigung CALL und RETURN und
– die E/A-Anweisungen READ, WRITE, PRINT, OPEN, CLOSE, INQUIRE (zur Abfrage des Dateizustandes). BACKSPACE (Positionieren auf den vorangegangenen Datensatz in der Datei), REWIND (Positionieren an den Dateianfang) und ENDFILE (Schreiben des Dateiende-Datensatzes).

Gegenüber FORTRAN IV bietet FORTRAN 77 folgende wesentliche Verbesserungen:

– Es sind Felder bis zu sieben Dimensionen erlaubt. Sowohl die untere als auch die obere Grenze jeder Dimension kann vereinbart werden
– mit dem Datentyp CHARACTER und dem Verkettungsoperator sind Zeichenkettenmanipulationen möglich
– mit der Block-IF-Anweisung wird die strukturierte Programmierung besser unterstützt
– zur Schleifensteuerung in DO-Anweisungen sind Ausdrücke erlaubt(bei entsprechenden Ausdruckwerten muß die DO-Schleife nicht unbedingt durchlaufen werden)
– für Konstanten können in PARAMETER-Anweisungen symbolische Namen vereinbart werden. Die PARAMETER-Anweisung ist eine Spezifikationsanweisung und hat die Form
PARAMETER (name = ausdruck , ...)
– mit der Spezifikationsanweisung IMPLICIT ist es möglich, die impliziten Typannahmen in Abhängigkeit von den Anfangsbuchstaben der Variablennamen zu ändern. Die Form der IMPLICIT-Anweisung ist IMPLICIT typname

(z_1 ... z_n), ... (z_i sind Anfangsbuchstaben)
– mit der Spezifikationsanweisung SAVE der Gestalt
SAVE name,"...
können in Unterprogrammen die Werte von Variablen, Feldern und benannten COMMON-Blöcken für die Nutzung beim nächsten Aufruf gerettet werden
– in numerischen Ausdrücken dürfen verschiedene Datentypen gemischt auftreten. Vor dem Ausführen von Zuweisungen und arithmetischen Operationen erfolgt gegebenenfalls eine implizite Typenkonvertierung
– mit einem impliziten DO in der DATA-Anweisung ist eine bequeme Dateninitialisierung möglich
– das erweiterte E/A-Konzept enthält neben dem sequentiellen Zugriff auch den Direktzugriff und neue Formatelemente
– der Satz von Standardfunktionen wurde wesentlich erweitert.

Der FORTRAN-77-Compiler für das SCP 1700 und das BOS 1810 erzeugt in mehreren Pässen Assemblercodes. Die Laufzeitbibliothek enthält Routinen zum Realisieren der E/A- und verschiedener mathematischer Funktionen. Zum Realisieren von Gleitkommaoperationen ist ein Gleitkommaprozessor oder -simulator erforderlich. Die Leistungen des Compilers können durch eine Reihe von Optionen eingestellt werden, mit denen der Programmierer Einfluß nehmen kann auf
– die Art und die Form der vom Compiler erzeugten Ausgaben (z. B. Quelltextliste, Crossreferenzliste)
– den Abbruch der Übersetzung nach bestimmten Fehlersituationen
– das Eingabeformat der Quellmoduln und
– den Umfang der durchzuführenden Codeoptimierungen.

Folgende Codeoptimierungen sind dabei im Compiler realisiert:

① Berechnen von schleifeninvarianten Ausdrücken und Felddadressen außerhalb von DO-Schleifen

② Einmaliges Berechnen von gleichen Teilausdrücken beim ersten Auftreten, wenn es die Datenflußeigenschaften des Programms gestatten und

③ Berechnen von konstanten Teilausdrücken zur Übersetzungszeit.

Der Compiler nutzt den Hauptspeicher optimal aus und ist auf einem A 7100 mit einer Konfiguration von 256 KByte Hauptspeicher, einem Bildschirmterminal und mindestens zwei Diskettenlaufwerken arbeitsfähig.

BASIC 1700

Die Programmiersprache BASIC wurde ursprünglich als Sprache für Anfänger geschaffen und orientiert deshalb auf die Entwicklung und das Abarbeiten von BASIC-Programmen im Dialog. Die Dialogfähigkeit von BASIC hat dazu geführt, daß für die meisten Personalcomputer BASIC-Interpreter und zum Teil auch BASIC-Compiler existieren. Der hohe Verbreitungsgrad der Sprache hat ihr Anwendungsgebiet erheblich erweitert. Heute wird BASIC zur Lösung wissenschaftlich-technischer Aufgabenstellungen und zum Bearbeiten kleinerer bis mittlerer Aufgaben auf dem Gebiet der ökonomischen Datenverarbeitung angewendet. BASIC 1700 besitzt eine einfache Syntax, die dem Anwender den schnellen und einfachen Einstieg in die Programmierung ermöglicht. BASIC 1700 basiert auf dem Minimal-BASIC-Standard (ANSI X3.60 – 1978) und erweitert diesen um eine Vielzahl von Sprachelementen. Die wichtigsten Erweiterungen sind:
– Außer Gleitkommadaten einfacher Genauigkeit (7 Stellen) werden auch Gleitkommadaten doppelter Genauigkeit (16 Stellen), Integerdaten (– 32768 bis 32767) und Zeichenketten bis zu einer Länge von 255 Zeichen unterstützt
– Identifikatoren von Variablen und Feldern können aus maximal 40 Zeichen bestehen. Damit wird die Lesbarkeit der BASIC-1700-Programme gegenüber anderen BASIC-Versionen wesentlich erhöht
– Felder können bis zu acht Dimensionen haben
– zum besseren Anwenden der strukturierten Programmierung können verschachtelbare IF-THEN-ELSE- und WHILE-WEND-Anweisungen benutzt werden
– in BASIC 1700 wird die satzweise und die wahlfreie Ein- und Ausgabe von Dateien sowie die formatgesteuerte Ausgabe (PRINTUSING-Anweisung) unterstützt
– mit Segmentierungsanweisungen (CHAIN) wird die Verkettung mehrerer Anwenderprogramme zu größeren Programmkomplexen möglich
– logische Operatoren sowie Oktal- und Hexadezimalkonstanten werden unterstützt.

Der BASIC-1700-Interpreter gestattet
– die Programmeingabe vom Bedienterminal oder von Dateien
– die zeilenweise oder in einem speziellen EDIT-Modus die zeichenweise Programmeditierung
– den Programmtest mittels TRACE- und Taschenrechneranweisungen
– die Programmabarbeitung und
– die Programmausgabe auf das Bedienterminal, den Drucker oder Dateien.

Für Anwenderprogramme werden vom Interpreter bis zu 64 KByte Hauptspeicher vorgesehen.

Standard-BASIC

Standard-BASIC entspricht dem ANSI-Standardisierungsvorschlag von 1984. Dieser Vorschlag geht vom ANSI-Standard X3.60 – 1978 aus und berücksichtigt bei allen Erweiterungen internationale Tendenzen auf den Gebieten der Programmiertechnologie und der Entwicklung und Anwendung der Programmiersprache BASIC. Demzufolge beschreibt der ANSI-Vorschlag nicht die Programmiersprache Standard-BASIC schlechthin, sondern das Programmiersystem Standard-BASIC. Der Vorschlag beinhaltet einen Standard-BASIC-Kern und sechs Erweiterungen. Der Kern umfaßt Ausdrucksmittel zur
– Hantierung von numerischen und Zeichenkettenwerten in Form von Konstanten, einfachen Variablen sowie ein- und zweidimensionalen Feldern einschließlich von Matrixoperationen
– Programmverzweigung und -strukturierung (GOTO-, GOSUB-, IF-, FOR-, DO- und SELECT-Aweisung)
– Realisierung der Unterprogrammtechnik und zur Programmverkettung

– sequentiellen Ein- und Ausgabe von Daten auf bzw. von verschiedenen Geräten bzw. Dateien und zur
– Ausnahmebehandlung und Testunterstützung.
Die Erweiterungen umfassen
– einen Editor
– numerische Werte in Festkommadarstellung speziell für ökonomische Anwendungen
– Erweiterungen der Ein- und Ausgabemöglichkeiten
– einen Grafikteil und
– einen Echtzeitteil.
In der ersten Ausbaustufe des Standard-BASIC-Programmiersystems werden der beschriebene Kern (mit Ausnahme der Matrixoperationen) und die Editor-Erweiterung als Compiler-Interpreter-System realisiert.
Als wesentliche Leistungen bietet dieses System
– die Ein- und Ausgabe von Programmen über das Bedienterminal und Dateien
– das Editieren von Programmen
– das Übersetzen von Programmen in einen Zwischencode mit einem Compiler
– das interpretative Abarbeiten und Testen von Zwischencodeprogrammen mit einem dialogfähigen Interpreter und
– das Erzeugen von Assemblerprogrammen aus den Zwischenprogrammen mit einem Codegenerator.
Genauere Angaben zum Ressourcenbedarf des Systems sind aufgrund des geplanten Entwicklungsablaufes noch nicht möglich.

PLM86

PLM86 ist eine höhere Programmiersprache, die speziell für den Einsatz auf Mini- und Mikrorechnern entwickelt worden ist. Sie ist sehr gut für die Systemprogrammierung geeignet. In dieser Eigenschaft ist sie integraler Bestandteil des BOS 1810 und weiterer Betriebssysteme für Mikrorechner. Darüber hinaus besitzt PLM86 gute Eigenschaften, die ihre Anwendung im wissenschaftlich-technischen und ökonomischen Bereich ermöglichen. Ebenso ist PLM86 bei der Implementierung komplexer Aufgaben der Prozeßsteuerung und Prozeßautomatisierung unter dem Betriebssystem BOS 1810 vorteilhaft einsetzbar. Die syntaktische Verwandtschaft von PLM86 mit PL/1 ist unverkennbar. Im Gegensatz zu PL/1 verfügt PLM86 aber über eine einfache Syntax und ist recht schnell und leicht erlernbar.
PLM86 unterstützt die Basisdatentypen
– BYTE (8-Bit-Größe)
– WORD (16-Bit-Größe)
– INTEGER (–32768 bis 32767)
– REAL (Wertebereich ist implementationsabhängig) und
– POINTER (Objektgröße ist vom Speichermodell abhängig).
Möglichkeiten der Datenstrukturierung sind mit der Deklaration von eindimensionalen Feldern und von Strukturen gegeben. Darüber hinaus werden die statische (AT-Attribut) und die dynamische oder pointerbezogene (BASED-Variable) Datenüberlagerung ermöglicht.
Mit Hilfe des INITIAL und des DATA-Attributes können in PLM86 Datenobjekte initialisiert werden. Für die Manipulation der Daten vom Basisdatentyp gibt es arithmetische, logische Relations- und Adreßoperatoren. Die strukturierte Programmierung wird von einem Prozedurkonzept und von geeigneten Anweisungen zur Programmablaufsteuerung unterstützt.
Das Prozedurkonzept sieht die call-by-value-Parametervermittlung vor. Sollen Werte über den Gültigkeitsbereich einer Prozedur hinaus weitergereicht werden, so sind die Adressen entsprechender Objekte (im übergeordneten Gültigkeitsbereich deklariert) der Prozedur als Eingangsparameter bereitzustellen. Entsprechend ihres Verwendungszweckes können in PLM86 Prozeduren mit zusätzlichen Attributen versehen werden, wie zum Beispiel
– INTERRUPT (für eine Interruptbehandlungsprozedur)
– REENTRANT (für rekursive Prozeduren) sowie
– PUBLIC und EXTERNAL (zum Festlegen des Gültigkeitsbereiches).
Neben Anweisungen zur Datendeklaration und zur Spezifikation von Parametern sowie externen Datenobjekten und Prozeduren verfügt PLM86 über folgende ausführbare Anweisungen:
– Die Zuweisung
– die GOTO-Anweisung
– die IF-THEN-ELSE-Anweisung
– vier verschiedene DO-Anweisungen (einfache DO-Anweisung, iterative DO-Anweisung, DO-WHILE-Anweisung und DO-CASE-Anweisung)
– die END-Anweisung zum Abschluß von komplexen Anweisungen und Prozeduren
– die CALL- und die RETURN-Anweisung
– die HALT-Anweisung sowie
– die ENABLE- und DISABLE-Anweisung für die Interruptbehandlung.
Des weiteren kann in PLM86 ein umfangreicher Satz von sogenannten built-in-Prozeduren zum codeeffektiven Ausführen vieler häufig wiederkehrender Teilaufgaben genutzt werden. So existieren zum Beispiel built-in-Prozeduren
– zum Ermitteln von Informationen über deklarierte Datenobjekte
– zur Typenkonvertierung
– zur Bitmanipulation
– zur Ein- und Ausgabe über E/A-Ports
– zur Zeichenkettenverarbeitung u.a.m.
Der PLM86-Compiler übersetzt PLM86-Quellmoduln in verschiebliche Objektcode. Als Übersetzungseinheit wird ein benannter einfacher DO-Block betrachtet. Der entstehende Objektcode kann mit den BOS-1810-Dienstprogrammen weiterverarbeitet werden (Linker, Locator, Bibliothekar). Die aus PLM86-Quelltext entstandenen Objektmoduln sind mit solchen Objektmoduln verbindbar, die aus Quellen anderer im BOS 1810 unterstützter Sprachen entstanden sind.
Darüber hinaus zeichnet sich der PLM86-Compiler durch folgende Eigenschaften aus:
– Es werden detaillierte Fehlernachrichten auch hinsichtlich der Typverträglichkeiten protokolliert
– der Compiler gestattet das bedingte Übersetzen und das Einbinden anderer Quelltextfiles (Include)
– der PLM86-Programmierer kann zwischen vier Speichermodellen (SMALL, COMPACT, MEDIUM und LARGE) wählen

– der PLM86-Compiler führt Codeoptimierungen in vier wählbaren Stufen aus
– besonders hervorzuheben ist die Fähigkeit, aus PLM86-Programmen sämtliche Prozeduren der Komponenten des Betriebssystems BOS 1810 direkt rufen zu können. Damit sind auch die Echtzeit- und Taskverwaltungsmöglichkeiten des BOS 1810 in PLM86-Programmen unkompliziert nutzbar. Daraus folgt, das PLM86 auch für den Einsatz bei der Prozeßautomatisierung gut geeignet ist.

MODULA-2

Die Programmiersprache MODULA-2 ist für Aufgaben der Systemprogrammierung sowie der Anwendungsprogrammierung gleichermaßen gut geeignet. MODULA-2 wird allgemein als Weiterentwicklung von PASCAL bezeichnet. In der Tat wurden viele Konstruktionen von PASCAL mit geringfügigen syntaktischen Änderungen in MODULA-2 übernommen. Einen wesentlichen Einfluß auf MODULA-2 hat aber auch MESA, eine Firmensprache von XEROX, ausgeübt. Dies betrifft insbesondere das leistungsfähige Modulkonzept, mit dem

– der Entwurf und die arbeitsteilige Entwicklung komplexer Programme
– das exakte Beschreiben der Modulschnittstellen und ihre Trennung von der Realisierung der Moduln
– das separate Compilieren der Moduln bei gleichzeitigem Überprüfen der Paßfähigkeit ihrer Schnittstellen
– das Mehrfachverwenden von Moduln
– die Programmierung paralleler Prozesse mittels eines einfachen Coroutinenkonzepts
– das Konzept der Datenabstraktion und
– die Bibliotheksarbeit

unterstützt werden.

Das Modulkonzept umfaßt vier Arten von Moduln:

– Den Programmodul
– den Definitionsmodul
– den Implementationsmodul und
– den inneren Modul.

Im Normalfall besteht ein MODULA-2-Programm aus mehreren Moduln. Der Programmodul ist dabei der Hauptmodul. Ein MODULA-2-Programm darf also nur einen Programmodul enthalten. Im Definitionsmodul wird die Schnittstelle eines Moduls (außer Programmodul) nach außen festgelegt, ohne das Verhalten des Moduls algorithmisch beschreiben zu müssen. Die Implementation der im Definitionsmodul beschriebenen Schnittstelle erfolgt im Implementationsmodul. Definitions- und Implementationsmodul sind selbständige Übersetzungseinheiten, bilden aber als Bestandteil eines MODULA-2-Programms eine Einheit. Innere Moduln können im Programmodul und in Implementationsmoduln auftreten und sind Hilfsmittel für die Datenkapselung und für die strukturierte Programmierung. Die Verbindungen zwischen den Moduln heißen Import-Export-Beziehungen und werden in den IMPORT- und EXPORT-Anweisungen beschrieben.

MODULA-2 bietet die gleichen Möglichkeiten der Konstanten-, Typ-, Daten- und Prozedurdefinition wie PASCAL. Hinzugekommen sind jedoch die Basisdatentypen CARDINAL (0...Maxcard), BITSET (entspricht SET OF [0...15]) und PROCEDURE sowie die Möglichkeit der Moduldeklaration. In einem Pseudomodul mit dem Namen SYSTEM sind darüber hinaus die Datentypen ADDRESS, WORD und PROCESS definiert. ADDRESS ist zu jedem Pointertyp kompatibel. Objekte vom Typ ADDRESS können in arithmetischen Ausdrücken verwendet werden. Damit ist in MODULA-2 (im Gegensatz zu PASCAL) die Möglichkeit der Adreßrechnung gegeben. Zur Manipulation der Basisdatentypen sind in MODULA-2 die von PASCAL her bekannten arithmetischen Operatoren, logischen Operatoren, Relationsoperatoren und Mengenoperatoren vorhanden.

An Anweisungen existieren:

– Zuweisung
– Prozedurrufe
– IF-THEN-ELSIF-ELSE-Anweisung
– REPEAT-UNTIL-Anweisung
– CASE-Anweisung
– WHILE-Anweisung
– FOR-Anweisung
– WITH-Anweisung
– LOOP-Anweisung (unbedingte Schleife)
– EXIT-Anweisung zum Verlassen von LOOP's und
– RETURN-Anweisung zum Verlassen von Prozeduren.

Ähnlich wie in PASCAL kann der MODULA-2-Programmierer einen umfangreichen Satz von Standard-Prozeduren nutzen, darunter auch solche zur Speicherverwaltung und zum Einrichten dynamischer Objekte. Hervorzuheben ist, daß Namen von vor- bzw. nutzerdefinierten Typen gleichzeitig Namen von Anpassungsprozeduren an den jeweiligen Typ sind.

Die Sprache MODULA-2 ist maschinen- und betriebssystemunabhängig. Maschinen- und betriebssystemabhängige Konzepte und Komponenten werden in MODULA-2 in Form von Standardmodulen bereitgestellt. Als Beispiele seien Moduln zum Unterstützen der Terminal-E/A und der File-E/A genannt. Auf diese Weise wird gesichert, daß mit MODULA-2 portable Programme bei gleichzeitiger Nutzung der Betriebssystemmöglichkeiten geschaffen werden können.

Alle hier kurz angedeuteten Eigenschaften von MODULA-2 sichern, daß die Sprache gleichermaßen gut für den Entwurf und die Implementierung von Software nach modernen programmiertechnologischen Prinzipien geeignet ist. Außerdem überprüft der Compiler die Übereinstimmung von Entwurf und Realisierung. Der MODULA-2-Compiler für das SCP 1700 garantiert diese Eigenschaft und liefert detaillierte Fehlernachrichten. Er erzeugt Objektmoduln, die mit solchen Objektmoduln koppelbar sind, die aus Quellprogrammen anderer im SCP 1700 unterstützter Sprachen entstanden sind. Der realisierte Sprachumfang stimmt mit dem vergleichbaren MODULA-2-Compiler überein und entspricht der Sprachdefinition von N. Wirth einschließlich der Ergänzungen vom Mai 1984.

C

Die Programmiersprache C wurde Anfang der 70er Jahre als Firmensprache von Bell Laboratories entwickelt und

verbreitete sich mit dem Betriebssystem UNIX als dessen „Muttersprache". Heute steht C nicht nur in den verschiedenen UNIX-Varianten, sondern auch in den meisten Mikrorechnerbetriebssystemen zur Verfügung.
C ist eine Systemprogrammiersprache. Sie kann aber auch für die Programmierung von Anwendersoftware genutzt werden. Indem sie dem Programmierer kaum Einschränkungen auferlegt, verbindet sie durch ihre Ausdrucksmittel die Vorteile der Assemblerprogrammierung mit den Möglichkeiten der Programmierung in komplexen strukturierten Anweisungen. Dadurch wurde C auch von vielen Assemblerprogrammierern akzeptiert und als ihre zukünftige Programmiersprache angesehen. C bietet Mittel zur Deklaration skalarer (integer-, float- und character-Variable) und strukturierter (Strukturen, Vereinigungen und Felder) Datenobjekte. Die Datenobjekte können mit Speicherklassenattributen wie automatic (im Keller), static (in einem statischen Bereich), extern (externe Objekte) und register (im Maschinenregister) versehen werden. Damit kann und muß der C-Programmierer Einfluß auf die Effektivität seiner Programme nehmen. C unterstützt die Arbeit mit Pointern. Zur Manipulation der Datenobjekte stehen eine Vielzahl von arithmetischen Operatoren, Relationsoperatoren, Inkrement- und Dekrementoperatoren, Bitoperatoren und Zuweisungsoperatoren bereit. In C ist ein einstufiges Prozedurkonzept und ein Blockkonzept realisiert. Für die Programmablaufsteuerung existieren die
– if-elseif-else-Anweisung
– switch-Anweisung, in der jeder Zweig eine case-Anweisung ist
– for-Anweisung
– while-Anweisung
– do-while-Anweisung
– break-Anweisung zum Verlassen von Switch-Zweigen und Schleifenanweisungen
– goto-Anweisung
– continue-Anweisung und
– return-Anweisung.
Die Ein- und Ausgabe, das Filemanagement, die Speicherverwaltung mit Hilfe Anwenderprogrammen sowie explizite Typkonvertierungen sind nicht Bestandteil der Sprache, sondern werden, mit Funktionen aus der umfangreichen C-Laufzeitbibliothek realisiert. Im C-Compiler für das SCP 1700 ist der Sprachvorschlag von Kernighan und Ritchie realisiert. Eine effektive Programmentwicklung in C wird durch Möglichkeiten der Quelltexteinschließung (include), der Makrodefinition und -benutzung auf C-Niveau und der bedingten Übersetzung unterstützt.

COBOL 1700

COBOL ist die Programmiersprache, die international am häufigsten insbesondere für kommerzielle Probleme und zur Massendatenverarbeitung eingesetzt wird. Sie wurde zwar ursprünglich für Großrechner entwickelt, hat sich aber inzwischen auch bei Mikrorechnern durchgesetzt. Dazu hat sicherlich auch die Standardisierung von ANSI (1974) und ISO (1978) beigetragen. COBOL unterstützt wie auch andere höhere Programmiersprachen die Deklaration und Manipulation numerischer Daten und die Strukturierung eines Programms durch Prozeduren. Entsprechend den Anwendungsgebieten existieren in COBOL vielfältige Ausdrucksmittel zum Beschreiben von Dateien, für den Zugriff zu den Dateien sowie der Manipulation der Dateien auf einem hohen problemnahen Niveau.
Der COBOL-1700-Compiler realisiert die Stufe 2 des ANSI-Standards von 1974. Damit wird die Quelltextkompatibilität zum COBOL 1630, COBOL 1620 und COBOL 1520 garantiert. Darüber hinaus werden im COBOL 1700 die interaktive Arbeit, der Aufruf von Codeunterprogrammen und Bibliotheksmöglichkeiten unterstützt. Der Compiler führt eine ausführliche Fehlerdiagnose aus.

PASCAL

Aufgrund der Verwandtschaft mit MODULA-2 und des hohen Verbreitungsgrades von PASCAL wird hier auf eine Erläuterung von Spracheigenschaften verzichtet.
Im PASCAL-Compiler für das SCP 1700 wird der ISO-Standard ISO-7185 – 1983, Niveau 0, realisiert. Der Sprachumfang ist damit identisch mit dem von PASCAL für das Betriebssystem MOOS 1600.

3.
Einige Bemerkungen zur Entwicklungstechnologie

Die Zielstellungen, die sich der VEB Robotron-Projekt Dresden hinsichtlich der Compilerentwicklung gesetzt hat, wurden im ersten Abschnitt ausführlich erläutert. Zum Realisieren dieser Zielstellungen ist es erforderlich, die Compilerentwicklung nach modernen softwaretechnologischen Prinzipien durchzuführen. Das betrifft einerseits das Anwenden höherer Programmiersprachen für die Phasen Entwurf und Implementation und andererseits die Nutzung von vorgefertigten und von Spezialprogrammen generierten Compilerbausteinen. Im VEB Robotron-Projekt wird deshalb zukünftig auf den Einsatz höherer Systemprogrammiersprachen wie MODULA-2 und C, und die Nutzung von Komponenten des Systems VINCENT (Verfahren und Instrumente zur Compilerentwicklung) zur Implementation von Programmiersprachen orientiert. Beim Kooperationspartner TU Dresden, Sektion Informationsverarbeitung, wird das CDL-Programmiersystem für die Compilerentwicklung eingesetzt.

Literatur
Nyderle, W.: Spektrum der auf den Rechnerklassen des VEB Kombinat Robotron implementierten höheren Programmiersprachen
Neue Technik im Büro, 4 (1985)
ANSI X3.9-1978: American National Standard Programming language FORTRAN
B.W. Kernighan, D.M. Ritcie: The C Programming Language
Prentice-Hall, Inc., Englewood Cliffs New Jersey 07632, 1978
N Wirth: MODULA-2
Second Edition, ETH-Berichte, Dezember 1980
ANSI X3J2/82: Draft Proposed American National
Standard for BASIC
15. Februar 1982
G. Paulin: MODULA-2-Report
In edv aspekte 3/85, S. 48–64

REDABAS – relationales Datenbankbetriebssystem für 8- und 16-Bit-Mikrorechner

Ursula Hempel, Hans Loley
VEB Robotron-Projekt Dresden

Leistungsfähigkeit und Flexibilität von REDABAS/M8 führten zu einer hohen Nutzerakzeptanz sowohl von qualifizierten EDV-Anwendern als auch von EDV-Laien. Die Anwendungsbereiche dieses Standard-Softwarepaketes reichen von der Industrie bis zur Medizin, von der Forschung bis zur Produktion. Dabei lernten vor allem Mitarbeiter ohne rechentechnische Spezialkenntnisse den Wert einer Datenbank als Grundlage für die Lösung komplexer technisch-organisatorischer Aufgaben zu schätzen. Die relativ leichte Handhabbarkeit und die Möglichkeiten der Programmierung zeichnen REDABAS als Datenbankbetriebssystem besonders aus.
In logischer Konsequenz wird REDABAS für 16-Bit-Rechner weiterentwickelt.
REDABAS/M16 ist – analog zur 8-Bit-Version – ein relationales Datenbankbetriebssystem mit einheitlicher Kommunikations- und Programmiersprache für Mikrorechner der 16-Bit-Klasse.
REDABAS/M16 läuft unter der Regie des Betriebssystems SCP 1700.

Voraussetzungen für die Anwendung von REDABAS/M16

Von einem REDABAS-Nutzer werden keine speziellen rechentechnischen Erfahrungen erwartet. Wer sich für eine rechnergestützte Erfassung, Verwaltung und Auswertung kleiner bis mittlerer Informationsmengen interessiert und über die erforderliche Hardware verfügt, kann REDABAS nutzen. REDABAS besitzt eine eigene, leicht erlernbare Kommandosprache, die – um einige Befehle erweitert – auch zum Erstellen von Anwendungsprogrammen genutzt werden kann.
Welche Hardware ist für die Anwendung von REDABAS/M16 erforderlich?

- 1 Arbeitsplatzcomputer ab 128 kBytes RAM (auf Basis Mikroprozessor K 1810 WM86 oder I8086), z. B. AC A 7100
- mindestens 1 Diskettenlaufwerk für 5,25"- oder 8"-Disketten
- 1 Bildschirm
- 1 Drucker (wahlweise).

Als Betriebssysteme werden

- SCP 1700 oder kompatible Betriebssysteme

gefordert.

Lieferumfang

REDABAS/M16 ist auf einer Diskette gespeichert und enthält folgende Programmkomponenten:

- REDABAS.CMD
- RDBASOVR.OVL
- RDBASMSG.TXT
- RDBASINS.CMD.

Zum Lieferumfang gehören außerdem Anwendungsbeschreibung und Programmtechnische Beschreibung.

Kurzcharakteristik von REDABAS

REDABAS ist ein Datenbankbetriebssystem, mit dem eine komplette Datenbank erstellt werden kann. Den Prinzipien der Datenbanktechnologie gemäß brauchen die Daten nur einmal eingegeben werden, stehen jedoch für unterschiedliche Einsatzmöglichkeiten und Anwendungsprogramme zur Verfügung. Der erforderliche Mehrfachaufwand für die Wartung von Einzeldateien entfällt. Änderungen von Daten und Anwendungsprogrammen sind problemlos möglich, wobei die Änderung an Daten kein Neuerstellen der Programme erfordert (Datenunabhängigkeit). Es können jederzeit Daten hinzugefügt, gelöscht, angezeigt und ausgedruckt werden. Für Auswertungen und Übersichten steht der Listengenerator zur Verfügung, wobei auch Multiplikation, Division, Teil- und Gesamtsummierung möglich sind.
Alle diese Funktionen werden über die REDABAS-Kommandosprache gesteuert, in der auch programmiert werden kann.
REBADAS verwaltet formatierte Daten und speichert die Nutzerdaten in Sätzen fester Länge. Der Anwender definiert den Datensatzaufbau, indem er für jedes Feld einen Namen, den Datentyp und die Länge festlegt.
Es können beliebig viele Datenbankdateien aufgebaut werden. REDABAS kann zwei Datenbankdateien gleichzeitig verarbeiten. Pro Datei sind max. 65 535 Sätze möglich. Ein Datensatz darf nicht größer als 1000 Zeichen sein, maximal sind 32 Felder möglich. Ein Feld darf die Anzahl von 254 Zeichen nicht überschreiten /1/.

REDABAS-Anwendung

Der Einsatz von Büro-, Personal- und Arbeitsplatzcomputern ist ein wichtiger Markstein in der Entwicklung und Anwendung der Datenverarbeitung. Erstmals wird es auch Mitarbeitern in Fachabteilungen von Betrieben und Instituten möglich, Aufgaben und Probleme ihres speziellen Fachgebietes direkt am Arbeitsplatz rechnergestützt zu bearbeiten und zu lösen. Im allgemeinen verfügen diese Mitarbeiter nicht über EDV-Kenntnisse. Die daraus resultierende Forderung nach vorgefertigter, nutzerfreundlicher Software erfüllt REDABAS als Datenbankbetriebssystem für Mikrorechner in hohem Maße. Davon zeugen zahlreiche Einsatzfälle für Büro- und Personalcomputer der 8-Bit-Klasse. Die Erfahrung mit REDABAS/M8 zeigt zunächst einmal den großen Bedarf, den es für dieses Standardsoftwarepaket zur Datenverwaltung gibt. Die Anwenderbetreuung zu REDABAS/M8 zeigt darüber hinaus, daß es natürlich speziell bei dem neuen EDV-Nutzerkreis trotz relativ ausführlicher programmtechnischer Beschreibung Probleme, zumindest jedoch Unklarheiten und Fragen geben kann.
Im Funktionsumfang und der Handhabung von REDABAS/M8 und REDABAS/M16 gibt es keine prinzipiellen Unterschiede. Leistungsdifferenzen zugunsten von REDABAS/M16 ergeben sich aufgrund der Anwendung der neuen 16-Bit-Computer, die sich gegenüber den 8-Bit-Computern u. a. durch die größere Verarbeitungsbreite, die höhere Operationsgeschwindigkeit und den größeren Operativspeicher auszeichnen. Im folgenden wird auf einige Funktionen und Möglichkeiten, die REDABAS bietet, etwas detaillierter eingegangen, um auch EDV-Laien und gelegentlichen Nutzern einen effektiveren Einsatz von REDABAS zu ermöglichen.

Ausgewählte Probleme

Das Ordnen von Dateien (Sortieren oder Indizieren?)

Zu den Vorzügen von REDABAS gehört, daß die Datensätze in ungeordneter Reihenfolge eingegeben werden können. Für viele Anwendungen wird es aber notwendig sein, die Daten zu ordnen. REDABAS bietet dafür den SORT- und INDEX-Befehl.

Der SORT-Befehl verschiebt die Datensätze, um die Datei nach einem beliebigen, vom Nutzer gewünschten Feld in auf- oder absteigender Reihenfolge zu sortieren (z. B. Sortierung von Namen). Allerdings kann in einem SORT-Befehl nur nach einem Datenfeld sortiert werden, das von Anwendung zu Anwendung unterschiedlich sein kann. Später zugefügte Datensätze erfordern jedes Mal einen relativ zeitaufwendigen Sortiervorgang. Auch das Wiederauffinden eines einzelnen Datensatzes dauert in einer sortierten Datei relativ lange, da die sortierte Datei sequentiell durchsucht werden muß.

Eine Alternative für das Verarbeiten einer Datei in einer bestimmten Sortierordnung ist das Indizieren von Dateien. Im Gegensatz zur Sortierung, bei der die Datensätze physisch umgeordnet werden, wird beim Indizieren eine Datei erzeugt, die lediglich ein Verzeichnis der zu sortierenden Feldinhalte und die Verweise auf den jeweils zugehörigen Datensatz enthält. Soll die Datenbankdatei für unterschiedliche Anwendungen nach verschiedenen Datenfeldern organisiert werden, kann für jedes gewünschte Datenfeld eine Indexdatei erstellt und genutzt werden. Je Datei können beliebig viele solcher Indexdateien angelegt werden. Neue Einträge zu einer Datei werden automatisch in die jeweils verwendete Indexdatei aufgenommen.

Während beim Sortieren nur ein Feldname als Sortierbegriff angegeben werden kann, ist beim Indizieren die Angabe eines Ausdrucks erlaubt, der z. B. mehrere Feldnamen durch Operatoren verknüpft (z. B. "feldname1 + feldname2" oder "feldname3/feldname4 * 100"). Das Indizieren bringt außerdem den Vorteil, daß ein Datensatz durch Angabe des Indexbegriffes angesprochen werden kann. REDABAS durchsucht die Datenbankdatei dann nicht, bis der gesuchte Satz gefunden ist, sondern benutzt dafür die Indexdatei und findet dort den direkten Verweis auf den Datensatz selbst.

Bezüglich des Speicherplatzbedarfs ist die Sortierung der Indizierung überlegen, da kein zusätzlicher Platz nötig ist, während für jede Indizierung eine Datei auf der Diskette angelegt wird. Die Größe einer Indexdatei hängt nicht nur von der Länge des Indexbegriffs und der Datensatzzahl, sondern auch von der Schlüsselfolge in der Datenbankdatei ab. REDABAS hält für das Unterbringen neu hinzukommender Datensätze Speicherplatz in variabler Größe frei.

Soll eine Datei logisch sequentiell verarbeitet werden, ist eine Sortierung sinnvoll, während für den wahlfreien Zugriff zu einzelnen Datensätzen über ein Ordnungskriterium die Indizierung erforderlich ist.

Der Zeitbedarf beim Sortieren hängt neben der Dateigröße auch von der Vorsortierung der Datenbankdatei ab, während beim Indizieren außerdem die Anzahl der verknüpften Ordnungskriterien maßgebend ist /2/.

Das Erstellen von Listen

Der REPORT-Befehl von REDABAS bietet vor allem Anwendern, die nicht programmieren wollen, die Möglichkeit, relativ unkompliziert Listen, Berichte, Tabellen, Übersichten zu erzeugen und auf dem Bildschirm oder Drucker auszugeben. Dies geschieht im Dialog: REDABAS bietet eine Reihe von vorprogrammierten Möglichkeiten – der Nutzer trifft seine Auswahl.

Nachstehend ein einfaches Beispiel für die problemlose Handhabung des Listengenerators. Im zweiten Beispiel wird die Möglichkeit gezeigt, eine Liste, deren Darstellung einige Besonderheiten aufweist, aus dem Inhalt von zwei Datenbankdateien zu erstellen.

Beim Beispiel 1 soll zum Erstellen eines Zwischenberichts die Datei "versand" ausgewertet werden (Abb. 1).

Der Dialog zur Listenerstellung wird dann mit dem Befehl REPORT aktiviert. Der Nutzer hat lediglich die Aufgabe, auf die von REDABAS gestellten Fragen (in Großbuchstaben gedruckt) seinen Vorstellungen vom Aussehen der Liste entsprechend zu antworten.

```
DATEISTRUKTUR FUER:     A:VERSAND .DBD
ANZAHL DER SAETZE:          00010
DATUM DER LETZTEN AENDERUNG: 00.00.00
DATENBANKDATEI IM PRIMAERZUGRIFF
FELD    NAME      TYP  LAENGE     DEZ
001     LIEFNR    C    004
002     KUNR      C    003
003     BETRAG    N    008        002
004     DATUM     C    008
** SUMME **            00024
```

Abb. 1 Struktur der Daten "versand"

```
: REPORT FORM liste1 TO PRINT
GEBEN SIE EIN, L=LINKER RAND, Z=ZEILEN/SEITE, B=SEITENBREITE: l=5
,z=65,b=80
WUENSCHEN SIE SEITENUEBERSCHRIFT? (J/N) :j
GEBEN SIE DIE SEITENUEBERSCHRIFT EIN: Zwischenbericht — Versand
ZWEIZEILIGE AUSGABE? (J/N) :n
WUENSCHEN SIE GESAMTSUMMEN? (J/N) :j
WUENSCHEN SIE ZWISCHENSUMMEN IN DER LISTE? (J/N) :n
SPALTE  BREITE, INHALT
001     10,liefnr
GEBEN SIE DIE SPALTENBEZEICHNUNG EIN: (Lieferantnummer
002     8,kunr
GEBEN SIE DIE SPALTENBEZEICHNUNG EIN: (Kunden-nummer
003     12,betrag
GEBEN SIE DIE SPALTENBEZEICHNUNG EIN: Betrag
WUENSCHEN SIE GESAMTSUMMEN? (J/N) :j
004     15,datum
GEBEN SIE DIE SPALTENBEZEICHNUNG EIN: (Datum
005 (ET)
```

Abb. 2 Dialog für die Gestaltung des Zwischenberichtes

Abb. 2 zeigt die Definition der Liste. Alle Angaben des Nutzers werden von REDABAS in einer Reportdatei gespeichert (hier z. B. "list1").
Bei den Nutzereingaben gibt es einige Besonderheiten zu beachten. So ist es zum Beispiel wichtig zu wissen, daß ein Semikolon in Zeichenreihen einen Zeilenumbruch bewirkt. Das ist eine Möglichkeit für die mehrzeilige Ausgabe von Datensätzen oder Überschriften. Im obigen Falle wurde z. B. auf diese Weise die Spaltenbezeichnung auf zwei Zeilen aufgeteilt.
Die Sonderzeichen "<" und ">" bewirken eine rechts- oder linksbündige Spaltenüberschrift; ohne diese Zeichen wird die Überschrift innerhalb der verfügbaren Spaltenbreite zentriert. Datum- und Seitenangabe erfolgen automatisch, sie können aber durch das Steuerwort PLAIN in der REPORT-Befehlszeile unterdrückt werden. Eine Voreinstellung der Überschrift ist mit Hilfe des Befehls SET HEADING TO <text> möglich. In <text> können mittels der Makroersetzung auch variable Angaben eingefügt werden.
Ist eine Änderung des Listenaufbaus gewünscht, löscht man zunächst die Reportdatei und aktiviert dann erneut den REPORT-Dialog. Eine von REDABAS erzeugte Reportdatei (Dateityp.DEF) kann aber auch mit einem Texteditor bearbeitet werden oder innerhalb von REDABAS mit MODIFY COMMAND verändert werden. Im Beispiel werden die Informationen aus zwei Datenbankdateien ("versand" und "adressen") mit dem Listengenerator aufbereitet, um eine Versandliste zu erzeugen (Abb. 4).
Die Lösung dieser Aufgabe erfolgt in zwei Schritten. Zunächst wird eine Befehlsfolge für das Aktivieren der beiden Dateien und die Ausgabe der gewünschten Liste erstellt (versli.prg – Diese Befehlsfolge wird dann mit DO versli zum Ausführungszeitpunkt aufgerufen). Danach erfolgt das einmalige Definieren der Listenstruktur im REPORT-Dialog (Abb. 5).
In der Befehlsfolge "versli" muß als erstes der Zugriff zu den beiden Dateien synchronisiert werden (REDABAS kann gleichzeitig auf zwei Dateien zugreifen und diese bearbeiten, indem zwei voneinander unabhängige Datenbereiche – Primär- und Sekundärbereich – verwaltet werden). Mit SET LINK ON wird der Satzzähler im Primär- und Sekundärbereich auf die gleiche Position gesetzt. Voraussetzung ist dabei, daß die Datensätze beider Dateien, deren logische Verbindung über die Kundennummer (Datenfeld "kunr") hergestellt wird, die gleichen Kundennummern in gleicher Reihenfolge haben müssen. Durch Erweiterung des REPORT-Befehls um die FOR-Klausel (p.kunr=s.kunr) wird vermieden, daß Informationen aus nicht zusammengehörigen Datensätzen (aus Primär- und Sekundärbereich) für die Erstellung der Liste als zusammengehörig betrachtet werden. Die Bedeutung des STORE-Befehls wird in der späteren Listendefinition klar. Mit CLEAR werden die Dateien geschlossen und die Speichervariable freigegeben.
Bevor im nächsten Schritt der Listendefinition (Abb. 6) der REPORT-Befehl gegeben wird, müssen wiederum beide Dateien aktiviert werden und die Variable "v1", die die Begrenzungsstriche der einzelnen Spalten enthält, erzeugt werden.
In der anschließenden mit REPORT FORM... initiierten Listenerstellung wurde u. a. folgendes Problem behandelt. Die gewünschte Zeilenlänge (Format A4) ist 65 Zeichen (b=65), die Summe der auszugebenden Zeichen einschließlich der Trennzeichen ist jedoch größer. Das bedeutet, daß die auszugebenden Informationen auf jeweils zwei Druckzeilen zu verteilen sind. Das wird durch die Unterbringung mehrerer Datenfelder in einer Druckspalte erreicht. REDABAS erlaubt nicht, als Spalteninhalt eine Feldfolge (feldname1, feldname2, ...) auszugeben, gestattet aber die Formulierung eines sogenannten Ausdrucks. So kann man beispielsweise mehrere Felder vom Typ Zeichenreihe durch den Operator "+" zu einer Zeichenreihe verbinden. Das Einfügen eines Semikolons ermöglicht eine zeilengerechte Abtrennung der Felder. Auf diese Weise wurden die Datenfelder "strasse", "ort" und "plz" zusammengefaßt, wobei die Spaltenbreite vom jeweils längsten Feld bestimmt wird. Das Komma nach "Name" und "Ort" wird möglich, indem die Leerstellen der Zeichenreihen durch die Funktion TRIM eliminiert wurden.
Numerische Felder kann man auf diese Weise übrigens nicht behandeln, das "+"-Zeichen würde eine Addition der Zahlenwerte bewirken. Hier muß man die STR-Funktion nutzen, mit der nu-

SEITE 00001
10.08.86

Zwischenbericht – Versand

Liefer-nummer	Kunden-nummer	Betrag	Datum
0268	5	11.70	10.05.86
6851	24	7.80	20.05.86
0221	44	0.00	
3420	78	24.30	10.6.86
0655	98	40.00	10.05.86
0654	111	21.00	10.05.86
6045	112	0.00	
7777	123	44.50	10.05.86
0732	234	78.65	19.6.86
0999	456	30.40	15.05.86
** SUMME **		258.35	

Abb. 3 Liste „Zwischenbericht-Versand" (Dialoggestaltung)

DATEISTRUKTUR FUER: A:ADRESSEN .DBD
ANZAHL DER SAETZE: 00010
DATUM DER LETZTEN AENDERUNG: 00.00.00
DATENBANKDATEI IM PRIMAERZUGRIFF

FELD	NAME	TYP	LAENGE	DEZ
001	KUNR	C	003	
002	NAME	C	009	
003	VORNAME	C	007	
004	PLZ	C	004	
005	ORT	C	010	
006	STRASSE	C	014	
007	RUF	C	007	
** SUMME **			00055	

Abb. 4 Struktur der Datei "adressen"

```
.MODIFY COMMAND versli
* Druck der Versandliste
SET LINK ON
USE adressen
SELECT SECONDARY
USE versand
STORE '|||' TO v1
REPORT FORM versli FOR p.kunr = s.kunr TO PRINT
CLEAR
RETURN
```

Abb. 5 Befehlsfolge "versli"

```
:SET LINK ON
:USE adressen
:SELECT SECONDARY
:USE versand
:STORE "!!!" TO v1
:REPORT FORM versli FOR p.kunr=s.kunr TO PRINT
 GEBEN SIE EIN, L=LINKER RAND, Z=ZEILEN/SEITE, B=SEITENBREITE:1=1
 ,b=65
 WUENSCHEN SIE SEITENUEBERSCHRIFT? (J/N) :j
 GEBEN SIE DIE SEITENUEBERSCHRIFT EIN: Versandliste;- - - - - - - - - -
 ZWEIZEILIGE AUSGABE? (J/N) :n
 WUENSCHEN SIE GESAMTSUMMEN? (J/N) :n
 SPALTE   BREITE INHALT
 001        3,kunr
 GEBEN SIE DIE SPALTENBEZEICHNUNG EIN: Knr;- - -
 002        1,v1
 GEBEN SIE DIE SPALTENBEZEICHNUNG EIN: !!!
 003        11,TRIM(name)+","+vorname
 GEBEN SIE DIE SPALTENBEZEICHNUNG EIN: (Name,,Vorname;- - - - - - -
 004        1,v1
 GEBEN SIE DIE SPALTENBEZEICHNUNG EIN: !!!
 005        17,strasse+","+TRIM(ort)+" "+plz
 GEBEN SIE DIE SPALTENBEZEICHNUNG EIN: (Strasse,,Ort, PLZ;- - - - - - -
 - - - - - - - -
 006        1,v1
 GEBEN SIE DIE SPALTENBEZEICHNUNG EIN: !!!
 007        4,liefnr
 GEBEN SIE DIE SPALTENBEZEICHNUNG EIN: Lfnr;- - - -
 008        1,v1
 GEBEN SIE DIE SPALTENBEZEICHNUNG EIN: !!!
 009        8,betrag
 GEBEN SIE DIE SPALTENBEZEICHNUNG EIN: )Betrag;;- - - - - - - - -
 010 (ET)
```

Seite 00001
10. 8. 1986

Versandliste

Knr	Name, Vorname	Strasse, Ort, PLZ	Lfnr	Betrag
5	Schiele, Christa	Reicker Str. 9 DRESDEN, 8036	0268	11.70
24	Schmidt, Heinz	Heckenweg 5 DRESDEN, 8046	6661	7.80
44	Schneider, Regina	Gartenstr. 24 RABENAU, 8222	0221	0.00
78	Marbert, Max	Dahlstr. 21 RADEBEUL, 8122	3420	24.30
98	Walther, Lilo	Weinberg 9 PESTERWITZ, 8211	0655	40.00
111	Preisel, Ekkart	Holbeinstr. 4 DRESDEN, 8019	0854	21.00
112	Guenzel, Klara	Am Anger 12 FREITAL, 8212	0045	0.00
123	Berlau, Herbert	Lindengasse 14 THARANDT, 8223	7777	44.50
234	Schmidt, Herbert	Am See 85 FREITAL, 8212	0732	78,65
456	Ober, Oswald	Kiestr. 10 DRESDEN, 8020	0999	30.40

merische Werte in eine Zeichenreihe umgewandelt werden können. Erst dann kann mit dem "+"-Operator verknüpft werden.

Für das Unterstreichen der Überschriften wurde die Steuerung des Semikolons für den Zeilenumbruch ausgenutzt. Im Programm „versli" wurde die Speichervariable "v1" angelegt. "v1" ist 3 Byte lang ("|||"). Im REPORT-Dialog wurde die Spaltenbreite für das Feld "v1" jedoch mit 1 definiert. Auf diese Weise erreicht man die Ausgabe der drei Striche untereinander. Gleichzeitig wird damit die Ausgabe einer Leerzeile für eine übersichtliche Darstellung der einzelnen Datensätze erreicht. Mit dem Befehl "DO versli" kann die in Abb. 7 gezeigte Liste jederzeit erzeugt werden.

Das Erzeugen von Datenbankkopien, das Übernehmen von Datenbeständen

Sicher gibt es Nutzer, die mit REDABAS erzeugte Datenbestände oder Auszüge davon mit in anderen Programmiersprachen geschriebenen Programmen oder Texteditoren weiterverarbeiten möchten oder umgekehrt, Datenbestände aus anderen Sprachen oder von Texteditoren übernehmen möchten. REDABAS unterstützt diese Form der Datenübergabe (Datenschnittstelle) mit den Befehlen COPY und APPEND. Ausgangspunkt ist das Standard-Daten-Format (SDF) des Betriebssystems SCP für Textdateien. Jede Textzeile ist dort mit den Zeilen für Wagenrücklauf und Zeilenvorschub (hex. 0D0A) abgeschlossen. Das bedeutet dementsprechend für das Dateiformat der Übergabedateien, daß jeder Datensatz durch <ET> und Zeilenvorschub (hex. 0D0A) oder nur <ET> begrenzt ist und das Dateiende durch „hex. 1A" gekennzeichnet ist. Die Datensätze müssen sequentiell organisiert sein und können feste oder variable Länge besitzen, der Dateninhalt muß aus druckbaren AS-CII-Zeichen bestehen.

COPY

Um eine übertragbare Datei (z. B. für die Bearbeitung mit einem Texteditor) aus einer REDABAS-Datenbankdatei zu erzeugen, wird der Befehl COPY in erweiterter Form verwendet. Nachfolgend werden vier Varianten des COPY-Befehls vorgestellt.

- COPY...SDF bewirkt, daß die Datenfelder in der definierten Länge ohne Begrenzungs- und Trennzeichen in der Textdatei angeordnet werden, wobei Leerstellen nicht entfernt werden. Als Beispiel sollen die ersten drei Datensätze der in Abb. 4 eingeführten REDABAS-Datenbankdatei "adressen" übertragen werden.

Mit einer Bereichsangabe und/oder einer Bedingung läßt sich die Anzahl zu übertragender Sätze einschränken. Sollen nicht alle Felder der Datei übertragen werden, kann man eine Feldauswahl treffen:

```
USE adressen
COPY TO dat1 NEXT 3 FIELD kunr,name,vor-
name,ort SDF
```

REDABAS überträgt die drei Datensätze mit den bezeichneten Feldern in

Abb. 6 Dialog für die „Gestaltung der Versandliste

Abb. 7 Versandliste – Gestaltung im Dialog

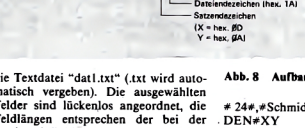


Abb. 8 Aufbau der Datei "dat1.txt"

die Textdatei "dat1.txt" (.txt wird automatisch vergeben). Die ausgewählten Felder sind lückenlos angeordnet, die Feldlängen entsprechen der bei der Strukturdefinition vergebenen Länge. Abb. 8 zeigt den Dateiaufbau der Textdatei (Mit dem SCP-Kommando TYPE dat1.txt oder einem Texteditor kann man sich die Daten anzeigen lassen).

- COPY...SDF DELIMITED bewirkt, daß alle Datenfelder im Ausgabesatz durch Kommas voneinander getrennt sind, Zeichenreihen in Hochkommas eingeschlossen sind und Leerstellen am Ende von Zeichenreihenfeldern entfernt werden.

Es sollen wiederum drei Datensätze der Datei „adressen" mit ausgewählten Feldern übertragen werden. Die Notation dieser Variante lautet:

```
COPY TO dat2 NEXT 3 FIELD kunr,name,vor-
name,ort SDF DELIMITED.
```

Der Aufbau der Textdatei ("dat2.txt") mit nunmehr variabel langen Sätzen sieht jetzt folgendermaßen aus:

```
' 5','Schiele','Christa','DRESDEN'XY
' 24','Schmidt','Heinz','DRESDEN'XY
'44','Schneider','Regina','RABENAU'-
XYZ
```

- COPY...SDF DELIMITED WITH (Begrenzungszeichen)

Diese Variante des COPY-Befehls ermöglicht die Auswahl eines Begrenzungszeichens. Es sind alle Sonderzeichen (außer < und >) erlaubt.

```
COPY TO dat3 NEXT 3 FIELD kunr,name,vor-
name,ort SDF DELIMITED WITH #
```

Die Textdatei "dat 3" ("dat3.txt") unterscheidet sich von "dat2" nur im Begrenzungszeichen der Datenfelder:

```
# 5#,#Schiele#,#Christa#,#Dres-
den#XY
# 24#,#Schmidt#,#Heinz#,#DRES-
DEN#XY
# 44#,#Schneider#,#Regina#,#RABE-
NAU#XYZ
```

- COPY...SDF DELIMITED WITH, bewirkt daß REDABAS nur die Felder in der Textdatei durch Kommas abtrennt und keine zusätzlichen Begrenzungszeichen für die Felder vom Typ Zeichenreihe vergibt. Führende Leerzeichen von numerischen Feldern werden entfernt ("dat4.txt").

```
COPY TO dat4 NEXT 3 FIELD kunr, name, vor-
name,ort SDF DELIMITED WITH,
  5,Schiele,Christa,DRESDENXY
 24,Schmidt,Heinz,DRESDENXY
 44,Schneider,Regina,RABENAUXYZ
```

Diese vier gezeigten Varianten demonstrieren vier unterschiedliche Möglichkeiten für den Aufbau einer Textdatei. Je nachdem, welche Datenstruktur das Verarbeitungsprogramm fordert, wird eine der Varianten gewählt. Es ist zu beachten, daß das Begrenzungszeichen nicht zum Datenfeldinhalt gehören darf. Kommen in Datenfeldern vom Typ Zeichenreihe Kommas vor, darf die Form "...DELIMITED WITH," nicht verwendet werden. Ein Komma im Datenfeld würde dann als Feldende gedeutet.

APPEND

Vorhandene Datenbestände können mit APPEND in eine REDABAS-Datenbank aufgenommen werden. Die Datenbankdatei muß definiert vorliegen. Enthält sie bereits Datensätze, werden die Datensätze aus der Textdatei angefügt. Prinzipiell gibt es zwei Möglichkeiten für die Datenstruktur in der Textdatei.

- APPEND...SDF

Die Datei hat eine feste Satzlänge, die der Summe der Feldlängen in der Datenbankdatei entspricht. Die Felder sind vollständig lückenlos, ohne Trenn- und Begrenzungszeichen und in der Reihenfolge entsprechend der Felddefinition der Datenbankdatei angeordnet. Jeder Datensatz muß mit <ET> und Zeilenvorschube (hex.0D0A) oder nur <ET> (= hex.0D) abgeschlossen sein, die Datei muß mit hex.1A abschließen. Soll beispielsweise die in Abb. 8 dargestellte Textdatei "dat1.txt" in eine Datenbankdatei übernommen werden, müssen folgende Befehle formuliert werden:

```
USE adressen
APPEND FROM dat1 SDF
```

Da keine einschränkende Bedingung formuliert wurde, übernimmt REDABAS alle Datensätze (im Beispiel nur drei Sätze) in die REDABAS-Datenbankdatei "adressen".

- APPEND...SDF DELIMITED REDABAS erwartet, daß die Datensätze in der Textdatei variabel lang sind und durch Kommas voneinander getrennt sind. Felder vom Typ Zeichenreihe können zusätzlich in Hochkommas oder Anführungszeichen eingeschlossen sein, andere Begrenzungszeichen werden als Feldinhalt interpretiert. Die Feldlängen von Text- und Datenbankdatei brauchen nicht übereinstimmen, kürzere Felder werden mit Leerzeichen aufgefüllt, längere werden abgeschnitten.

Ein Beispiel hierfür wäre die Übernahme von "dat2.txt" in die Datei "adressen". Der APPEND-Befehl lautet in diesem Fall

```
APPEND FROM dat2 SDF DELIMITED
```

Auch hier gilt, daß Begrenzungszeichen nicht Inhalt von Datenfeldern sein dürfen und daß der Feldinhalt dem bei der Definition der Dateistruktur festgelegten Datentyp entsprechen muß.

Auch beim APPEND-Befehl können Bedingungen für die Auswahl zu übertragender Datensätze formuliert werden.

Das Umbenennen von Datenfeldern

In der Praxis macht es sich mitunter erforderlich, Namen von Datenfeldern in einer REDABAS-Datenbankdatei zu

verändern. Mit dem Befehl APPEND FROM werden bekanntlich Daten von einer Datei in eine andere übertragen. Es werden jedoch nur die Datenfelder übertragen, deren Namen übereinstimmen. Mit der vom vorigen Abschnitt bekannten Erweiterung des APPEND-Befehls SDF ist es möglich, Namensveränderungen vorzunehmen. Soll beispielsweise das Feld "kunr" der Datei "adressen" in "schl" umbenannt werden, ist folgende Befehlsfolge erforderlich:

```
USE adressen
COPY TO temp SDF
MODIFY STRUCTURE*
APPEND FROM temp.txt SDF
```

* (Nach Anzeige der Dateistruktur wird das Feld "kunr" in "schl" geändert)

Bearbeiten geteilter Datenbestände

Mitunter ist die Kapazität einer Diskette nicht ausreichend, um alle für eine Datenbankdatei vorgesehenen Nutzerdaten aufnehmen zu können. Das Betriebssystem SCP und damit auch REDABAS lassen die Ausdehnung einer Datei über mehrere Disketten nicht zu. REDABAS unterstützt jedoch den Diskettenwechsel während einer Sitzung durch den RESET-Befehl. Legt man den ersten Teil einer Datenbankdatei auf der Diskette an und den zweiten Teil unter dem gleichen Dateinamen und mit der gleichen Dateistruktur auf einer zweiten Diskette, so kann z. B. ein einmal aktiviertes Verarbeitungsprogramm beide Dateien nacheinander mit derselben Befehlsfolge verarbeiten. Im folgenden dazu auszugsweise ein REDABAS-Anwendungsprogramm (datver. prg) für die Verarbeitung eines geteilten Datenbestandes (daten.dbd):

```
: MODIFY COMMAND datver
* Verarbeitung eines geteilten Datenbestandes
INPUT, 'Dateianzahl' TO v1
STORE, '1' TO v2
DO WHILE v1 > 0
USE b:daten
•
•
• Verarbeitungsbefehle
•
•
•
USE
STORE v1-1 TO v1
IF v1 0
STORE STR(VAL(v2)+1,1) TO v2
? 'Bitte Diskette', v2, 'einlegen und <ET> druecken!'
WAIT
RESET B
ENDIF
ENDDO
```

RETURN

Bei Abarbeitung der Befehlsfolge wird zunächst der Bediener nach der Anzahl der Dateien bzw. der Disketten gefragt, die den gesamten Datenbestand ausmachen. In der DO-WHILE-Schleife erfolgt die Bearbeitung der Datenbankdatei. Dieser Zyklus wird für jede Datei wiederholt und endet mit dem Abschließen der Datenbankdatei durch USE. Auf die Verarbeitungsbefehle folgt an den Bediener die Aufforderung, die nächste Diskette einzulegen, sofern der mitgeführte Zähler (v1) noch nicht den Wert 0 erreicht hat. Nach dem Diskettenwechsel wird der RESET-Befehl ausgeführt. In diesem Befehl sollte stets das Laufwerk angegeben werden, auf dem sich die Datenbankdatei befindet. Zu beachten ist ferner, daß die REDABAS-Programmdiskette nicht ausgewechselt werden darf. Nach Verarbeitung der letzten Diskette ist das Programm beendet. Natürlich müssen die Namen der Datenbankdateien nicht auf allen Disketten gleich sein. Man könnte z. B. die Dateien fortlaufend numerieren (daten1, daten2, daten3, ...) und den Dateinamen durch Anhängen des Zählers v2 an den konstanten Teil des Dateinamens (daten) im USE-Befehl generieren (USE b:daten&v2).

Literatur

/1/ Hempel, U.; LoLey, H.: Das relationale Datenbankbetriebssystem REDABAS. rechentechnik/datenverarbeitung 22(1985)11

/2/ Weber, P.: REDABAS – Laufzeitverhalten von Kommandos, Laufzeitgestaltung von Anwenderprogrammen. rechentechnik/datenverarbeitung 23(1986)7

/3/ Hempel, U.; Loley, H.: Datenbanken mit Personalcomputern. Verlag Die Wirtschaft Berlin, 1986

Neue Zeitschrift im Akademie-Verlag Berlin:

Ab Januar 1988 erscheint im Akademie-Verlag Berlin quartalsweise eine neue internationale Fachzeitschrift

Journal of new Generation Computer System

herausgegeben im Auftrage des Koordinierungsrates für Rechentechnik und Informatik der Akademien der Wissenschaften der sozialistischen Länder, am Zentralinstitut für Kybernetik und Informationsprozesse der Akademie der Wissenschaften der DDR.

Ziel und Profil

Journal of new Generation Computer Systems (JNGCS) dient der Veröffentlichung von Beiträgen, die der Entwicklung von Rechnersystemen mit qualitativ neuen Eigenschaften und neuartigen Anwendungsgebieten, d. h. der Schaffung neuer Generationen von Rechnersystemen gewidmet sind.

JNGCS berichtet über Fortschritte, Erfahrungen und neueste Ergebnisse bei der Realisierung nationaler und internationaler Forschungs- und Entwicklungsprogramme sowie von Projekten, speziell von Gemeinschaftsprojekten der sozialistischen Länder, deren Ziel die Schaffung innovativer Rechnersysteme ist.

Das Hauptziel der Zeitschrift ist der Informationsaustausch zu neuesten Entwicklungen der Hard- und Software sowie zu Anwendungen auf dem Gebiet der Verarbeitung von Wissen und großen Datenmengen.

Aus dieser Charakteristik ergibt sich für JNGCS ein Profil mit folgenden Interessengebieten:

- Höchstintegrierte Schaltkreise (VLSI)
- neue Entwurfs- und Fertigungstechnologien für Rechnersysteme
- innovative Architekturen für verschiedene Anwendungen
- Softwaretechnik mit künstlicher Intelligenz
- Wissensverarbeitung
- Mensch-Maschine-Kommunikation
- Theorie und Künstliche-Intelligenz-Forschung

Beiträge, die Forschungsergebnisse oder neue Technologien vorstellen, Übersichtsarbeiten, Berichte und Mitteilungen sind in Englisch und Russisch zu richten an:
Dr. W. Kolbe, Redaktion „JNGCS", ZKI/AdW, Kurstr. 33, PF 1298, Berlin, 1086, DDR.

Bestellungen der JNGCS nehmen entgegen: Akademie-Verlag Berlin, Leipziger Str. 3/4, PF 1233, Berlin, 1086

TEXT 40/M16 – Programmsystem zur Textverarbeitung auf 16-Bit-Bürocomputern

Rainer Weber
VEB Robotron-Projekt Dresden

Die immer breitere Anwendung der Rechentechnik, insbesondere von Büro- und Personalcomputern schafft die Voraussetzung für eine komplexe Rationalisierung einer Vielzahl von Schreib- und anderen Büroarbeiten. Das Programmsystem „Textverarbeitung auf 16-Bit-BC – TEXT 40" stellt ein solches Rationalisierungsmittel dar. Mit der Anwendung von TEXT 40 lassen sich vielfältige Aufgaben wie

– Texte erfassen, korrigieren, aktualisieren, mischen bei gleichzeitiger Formatierung einschließlich automatischer Silbentrennung
– Schreiben von Serienbriefen (z. B. Einladungen) auf der Grundlage eines einheitlichen Brieftextes und einer Namens- bzw. Adreßliste
– Terminüberwachung und -kontrolle

auf ein Minimum an Arbeitsaufwand für den Nutzer reduzieren. Das Erstellen von Dokumentationen wird durch Funktionen zur Erzeugung von Verzeichnissen und Registern unterstützt.
Das Programmsystem TEXT 40 ist für den Einsatz auf AC A 7100 und ähnlichen Büro- und Personalcomputern vorgesehen.
Durch die Programmierung in C ist eine hohe Portabilität und Hardware-Unabhängigkeit gewährleistet.

Anwendungsgebiete

Das Textverarbeitungssystem TEXT 40 ist unabhängig von spezifischen Anwendungsgebieten. Es kann überall dort eingesetzt werden, wo Texte oder Programme zu bearbeiten, d. h. zu erfassen, zu korrigieren, extern zu speichern, zu überarbeiten, zu gestalten, zusammenzusetzen sowie in bestimmter Form auszudrucken sind. Mit TEXT 40 können alle im Büro anfallenden Schreibarbeiten erledigt und zum Teil erheblich rationalisiert werden. Aber auch größere Schriftstücke wie z. B. Dokumentationen können mit TEXT 40 erfaßt, gewartet und ständig auf dem aktuellen Stand gehalten werden ohne dafür größere Schreibkapazität zu binden. Bei Bedarf können beliebig viele Exemplare von Schriftstücken oder Teile von ihnen (Ergänzungen, Änderungsblätter) kurzfristig hergestellt werden.

Verträglichkeit mit anderen Textverarbeitungssystemen

Mit TEXT 40 können Texte, die unter anderen Textverarbeitungssystemen erfaßt bzw. erarbeitet wurden, weiterverarbeitet werden. Voraussetzung dafür ist, daß auf diese Textdateien durch das Betriebssystem, unter dem TEXT 40 läuft, zugegriffen werden kann. Um dies zu erreichen, existieren bereits verschiedene Dateikonvertierungsprogramme, die nicht Bestandteil von TEXT 40 sind. Bei Bedarf können diese in das Textsystem TEXT 40 integriert werden, so daß mittels einfacher TEXT-40-Kommandos z. B. Text-20 – oder WordStar – Textdateien unmittelbar verarbeitet werden können.

Voraussetzung für die Anwendung

- Gerätetechnik

TEXT 40 ist an keine spezielle Hardware gebunden. Die Grunderfordernisse bezüglich Gerätekonfiguration sind:
– Zentraleinheit mit mindestens 128 kByte Hauptspeicher
– Externspeicher (Diskette oder Platte)
– Bildschirm
– Tastatur
– Drucker.

Das Programmsystem TEXT 40 ist so konzipiert, daß Hardware-Schnittstellen (Tastatur, Bildschirm, Drucker) mittels spezieller, austauschbarer Moduln realisiert werden, wodurch eine schnelle, problemlose Anpassung an beliebige Hardware-Gegebenheiten möglich wird. Dabei muß z. B. das Fehlen von Funktions- oder Cursorpositioniertasten auf der Tastatur mit Hilfe eines Mehraufwands bei der Bedienung (Eingabe eines Kommandos anstatt Betätigung einer Funktionstaste) kompensiert werden.
Durch den modularen Aufbau von TEXT 40 kann der Funktionsumfang den konkreten Wünschen eines Nutzers und seiner gerätetechnischen Möglichkeiten angepaßt werden.

- Spezialkenntnisse

Für die Nutzung von TEXT 40 sind keinerlei Spezialkenntnisse erforderlich. Der Anwender muß lediglich auf seinem Rechner ein Programm (nämlich TEXT 40) starten können. Die weitere Arbeit in TEXT 40 läuft menügesteuert, d. h. aus einer Liste von Funktionen kann die gewünschte ausgewählt werden. Die innerhalb einer solchen Funktion verfügbaren Kommandos und Funktionen sind als Gedankenstütze einer Informationszeile dem Bildschirm zu entnehmen. Zu allen Funktionen und Kommandos können mittels einer speziellen Hilfe-Funktion zusätzliche Informationen angefordert werden.

Funktionen von TEXT 40

Funktionskomplexe
Zur Erhöhung der Übersichtlichkeit und Vereinfachung der Bedienung wurde die Vielzahl der in TEXT 40 enthaltenen Funktionen in einzelne, leicht überschaubare Funktionskomplexe zusammengefaßt.

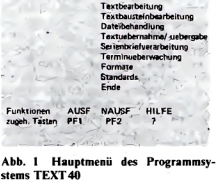


Abb. 1 Hauptmenü des Programmsystems TEXT 40

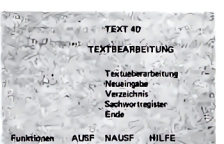


Abb. 2 Auswahl der benötigten Funktionen aus dem Menü

Wird das Programmsystem TEXT 40 gestartet, so erscheint auf dem Bildschirm das Hauptmenü (Abb. 1), aus dem man einen der Funktionskomplexe auswählen kann. Nach erfolgter Auswahl wird durch das zentrale Steuerprogramm der entsprechende Programmkomplex vom Externspeicher in den Hauptspeicher geladen. Dieser Programmkomplex enthält alle die Programme, die zur Realisierung aller Funktionen des angewählten Funktionskomplexes erforderlich sind und sich nicht im Systemkern befinden.

Ist der Programmkomplex geladen, erscheint auf dem Bildschirm das zu diesem Funktionskomplex gehörige Menü, aus dem wiederum die gewünschte Funktion ausgewählt werden kann (Abb. 2).

Allgemeingültige Funktionen

Neben Funktionen, die nur in einzelnen Menüzweigen verfügbar sind, gibt es auch solche, die in mehren oder allen Zweigen erforderlich sind. Welche Funktionen wo anwendbar sind, geht aus der Beschreibung der Menüzweige hervor.

■ **Cursorfunktionen**

– Positionieren

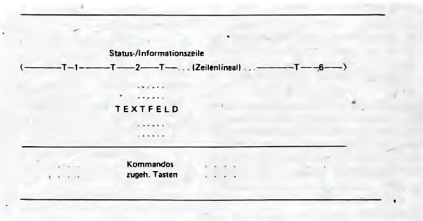
Mit den Cursorfunktionen "→", "←", "↑", "↓" kann der Cursor innerhalb des Textfeldes (Abb. 4) auf dem Bildschirm frei (in der jeweils gewünschten Richtung) bewegt werdene. Mit den Cursorfunktionen " ", " " kann der Cursor auf die erste beziehungsweise letzte Schreibposition der Seite gebracht werden.

– Rollen horizontal

Steht der Cursor am linken (beziehungsweise rechten) Rand des Textfeldes (des Bildschirm-Textfensters), so wird durch Betätigen der Cursorpositionierfunktion "→" bzw. "←" der Text um eine Bildschirmhälfte nach links bzw. rechts verschoben, sofern das Zeilenende bzw. der Zeilenanfang nicht bereits auf dem Bildschirm sichtbar sind.

– Rollen vertikal

Steht der Cursor in der letzten bzw. ersten Zeile des Textfensters und wird die Cursorfunktion "\|/" bzw. "/|\" betätigt, so wird der angezeigte Text um eine Zeile nach oben bzw. unten geschoben,



wobei die oberste bzw. unterste Zeile verschwindet. In der frei gewordenen untersten bzw. obersten Zeile wird die entsprechende Textzeile dargestellt.

Bei Erreichen des Seitenendes (des Seitenanfangs) wird eine Markierung des Seitenendes, die Seitennummer der Folgeseite (der aktuellen Seite) und die nächsten Zeilen der Folge- (der Vorgänger-)seite durch weiteres Betätigen der Funktionen "\|/" ("/|\") erscheinen.

– Bildschirmblättern vorwärts/rückwärts

Da eine Textseite oft mehr Zeilen enthält, als gleichzeitig auf dem Bildschirm dargestellt werden können, sind zwei Funktionen zum Bildschirmblättern realisiert. Mit ihrer Hilfe wird ein schnelles vertikales Rollen ermöglicht, so daß nach Betätigen der Funktion „nächste BS-Seite" die gerade noch letzte Zeile des Textfensters zur ersten Zeile des neuen angezeigten Textausschnittes wird.

– schnelles Blättern

Bei Aufruf der Funktion „schnelles Blättern" wird nach Angabe der erwünschten Seitennummer auf diese Textseite „geblättert" und die ersten Zeilen dieser Seite werden im Textfenster des Bildschirmes angezeigt.

■ Löschen

Mit der Funktion „LÖSCHEN" kann sowohl ein einzelnes Zeichen, das durch den Cursor markiert wird, als auch ein beliebig großer Bereich gelöscht werden.

■ Darstellungsfunktionen

Abb. 4 Eingabe des gewünschten Formates – Arbeitsmodi: EINGABE, FLATTERSATZ, SILBENTRENNUNG

Zu den Darstellungsfunktionen gehören:

– Unterstreichen
– Fettschrift
– Schattenschrift
– Sperrschrift
– Sparschrift
– Exponent
– Index.

Diese Funktionen sind sowohl auf Einzelzeichen, als auch auf Bereiche innerhalb des Textes anwendbar.

Funktionskomplex Textbearbeitung

Der Funktionskomplex Textbearbeitung ist der Hauptteil des Programmsystems TEXT 40. Innerhalb der Textbearbeitung können Texte erfaßt, korrigiert, umgestellt, formatiert und gestaltet werden. Innerhalb der Textbearbeitung gibt es verschiedene Arbeitsmodi, die zu beliebiger Zeit während der Arbeit umgestellt werden können. Des weiteren existieren neben den bereits beschriebenen allgemeingültigen Funktionen eine Reihe Gestaltungsfunktionen.

Textüberarbeitung

In der Bedienerführung (siehe Abb. 3) ist der Name der zu bearbeitenden Textdatei einzugeben oder aus der Liste auszuwählen. Auf dem Bildschirm erscheinen die Informationen entsprechend Abb. 4, wobei im Textfeld der Anfang

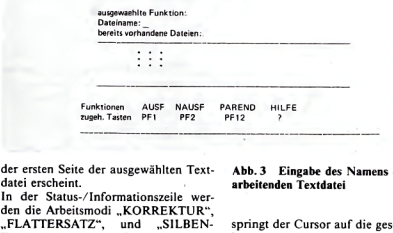


Abb. 3 Eingabe des Namens der zu bearbeitenden Textdatei

der ersten Seite der ausgewählten Textdatei erscheint.

In der Status-/Informationszeile werden die Arbeitsmodi „KORREKTUR", „FLATTERSATZ", und „SILBENTRENNUNG" angezeigt, sofern diese Standards nicht über den Menüzweig „STANDARDS" verändert wurden.

Neueingabe

Nach Auswahl der Funktion „NEUEINGABE" erscheint eine Bedienerführung, ähnlich der in Abb. 3, die zusätzlich zur Eingabe eines Dateinamens noch die Angabe des gewünschten Formates ermöglicht, falls nicht das Standardformat (A4) verwendet werden soll. Das entsprechend Abb. 4 erscheinende Bild enthält ein leeres Textfeld, die Arbeitsmodi sind „EINGABE", „FLATTERSATZ", „SILBENTRENNUNG" (entsprechend der definierten Standards).

Gestaltungsfunktionen

Zu den Gestaltungsfunktionen gehören:

- Tabulation
- Dezimaltabulation
- Zentrieren
- Trennen manuell
- Formatieren
- Suchen/Austauschen
- Umstellen
- Mischen
- Titel
- Untertitel
- Fußnoten.

● Tabulation

Die Tabulation ist ein Hilfsmittel beim Schreiben von Tabellen, Aufstellungen usw. Neben den im Format definierten Tabulatoren, die bei jeder Textverarbeitung verfügbar sind, können während der Bearbeitung zusätzliche Tabulatoren lokal definiert beziehungsweise gelöscht werden.

Die Ausführung der Tabulation erfolgt mit der Funktion „Tabulatorsprung" (→|). Bei Betätigen dieser Funktion springt der Cursor auf die gesetzten Tabulatorpositionen von links nach rechts, beginnend bei der nächsten gesetzten Position rechts des Cursors in der aktuellen Zeile.

● Dezimaltabulation

Die Dezimaltabulation ermöglicht die dezimalstellengerechte Eingabe von Zahlen und das rechtsbündige Schreiben von Zeichenfolgen in entsprechenden Tabellen. Die Ausführung kann an jeder beliebigen Tabulatorposition erfolgen, indem diese nicht mit der gewöhnlichen Tabulatorsprung-Funktion, sondern mit „DEZITAB" angesprungen wird.

Alle folgenden eingegebenen Zeichen werden bei jeder neuen Zeicheneingabe um eine Stelle nach links verschoben, der Cursor bleibt dabei auf der Tabulator-Position (Einerstelle) stehen. Die Funktion wird durch das Kommando „AUSF" oder ein Komma als Eingabezeichen abgeschlossen.

● Zentrieren

Die Funktion „Zentrieren" bewirkt, daß der Text der aktuellen Zeile in die Mitte gerückt wird.

● Trennen

Da der automatische Silbentrennalgorithmus mit einer Wahrscheinlichkeit von 98 Prozent richtig trennt, kommt es vor, daß fehlerhafte oder unpassende Trennungen rückgängig gemacht und durch manuelle Trennung ersetzt werden müssen. Dazu wird der Cursor auf das erste Zeichen der abzutrennenden Silbe gesetzt und die Funktion „TRENN" ausgelöst. Diese Funktion bewirkt, daß vorherige Trennungen in diesem Wort rückgängig gemacht werden und der bereits eingegebene Text von der aktuellen Zeile bis Absatzende neu formatiert wird.

● Formatieren

Mit dieser Funktion kann der Text ab Cursorposition bis zum Absatzende neu formatiert werden.

● Suchen/Austauschen

Die Funktion Suchen/Austauschen realisiert das Suchen einer beliebigen, als Suchbegriff einzugebenden Textstelle von der Cursorposition ab bis zum Textende (Dateiende). Bei jedem Auftreten des Suchbegriffs wird (über Bildschirm) gefragt, ob dieser gegen den Austauschbegriff zu ersetzen ist. Mit den Kommandos „AUSF" bzw. „NAUSF" wird das Austauschen realisiert bzw. unterdrückt. Daraufhin erscheint die Frage „Weitersuchen?", die mit „AUSF" bzw. „NAUSF" beantwortet werden kann, was einer Fortsetzung bzw. dem Abbruch der Funktion entspricht.

● Umstellen

Die Funktion „UMSTELLEN" dient dazu, die Reihenfolge beliebiger Textpassagen, Sätze, Absätze usw. ändern zu können.

Im Gegensatz zur Funktion „MISCHEN", wo Kopien von Textstellen ab Cursorposition eingetragen werden, der Texturspung dabei jedoch nicht verändert wird, bedeutet „UMSTELLEN", daß eine Textpassage aus seiner Umgebung herausgelöst und an die mittels Cursor bezeichnete Position übertragen wird.

„UMSTELLEN" ist nur innerhalb der in Bearbeitung befindlichen Textdatei möglich.

Vor Aufruf der Funktion „UMSTELLEN" steht der Cursor auf dem Zeichen, vor dem der umzustellende Text eingefügt werden soll. Nach dem Funktionsaufruf wird mittels Cursor der Bereich des umzustellenden Textes markiert (Abschluß mittels „PAREND"). Die Ausführung der Umstellung wird mittels „AUSF" eingeleitet.

● Mischen

Mit der Funktion „MISCHEN" können beliebige Texteinheiten, auch aus anderen Dateien in den laufenden Text eingefügt werden, ohne daß ihre ursprüngliche Umgebung verändert wird. Der Cursor steht wie beim Umstellen auf dem Zeichen, vor dem der einzumischende Text zu postieren ist.

Nach Funktionsaufruf kann zunächst ein Dateiname eingegeben werden, aus der die Texteinheit geholt werden soll. Wird dieser Name weggelassen, wird die aktuell in Bearbeitung befindliche Datei benutzt. Als nächstes werden Anfang und Ende des zu kopierenden Bereiches (jeweils mit „PAREND" markiert). Mit dem Kommando „AUSF" wird das Mischen realisiert.

- Titel

Eine Titelzeile kann zu Beginn der Texteingabe beziehungsweise Überarbeitung auf die entsprechende Aufforderung hin eingegeben werden.

- Untertitel

Innerhalb eines Textes können beliebige Textstücken (z. B. Kapitelüberschriften) als Untertitel gekennzeichnet werden. Die Kennzeichnung erfolgt mittels der Funktion „UNTERTITEL". Der auf diese Weise markierte Text wird von der nächsten Seite an auf jeder Seite als Untertitel geschrieben, bis durch erneute Untertitel-Definition dieser Text durch einen neuen ersetzt wird.

- Fußnoten

Fußnoten werden im laufenden Text eingegeben und mit der Bereichsfunktion „FUSSNOTE" gekennzeichnet. Bei der späteren Seitenstrukturierung wird dieser Fußnotentext durch einen Verweis auf die Fußnote ersetzt (hochgestellte Zahl und schließende runde Klammer), der Fußnotentext selbst erscheint in den letzten Zeilen der Seite, eingeleitet durch die gleichen Zeichen.

Funktionskomplex
Textbausteinbearbeitung

Bausteinerzeugung/-korrektur

Die Herstellung, Korrektur und Änderung von Textbausteinen unterscheidet sich nicht von der Textbearbeitung. Textbausteine sind ebenfalls Dateien, sie werden jedoch getrennt von den Textdateien gespeichert. Textbausteine sollten möglichst kurz sein, unterliegen jedoch keinen Beschränkungen.

Bausteinhandhabung

Textbausteine können bei der Texteingabe/-überarbeitung direkt eingesetzt werden, oder sie werden durch ihren Namen (Dateiname des Bausteins) im Text repräsentiert und zu einem späteren Zeitpunkt durch den Baustein ersetzt.

Der Name eines Textbausteines wird mit Hilfe des Sonderzeichens "#" im Text gekennzeichnet. Dabei bedeutet "#name", daß der unter diesem Namen existierende Baustein entsprechend dem gewählten Format eingefügt wird, sofern das Format des Textes dies zuläßt; andernfalls erscheint eine Fehlermitteilung. Bei der Eingabe einer Bausteinbezeichnung (Sonderzeichen und Name) kann diese wie gewöhnlicher Text eingegeben werden, was eine spätere Bausteinsubstitution (Ersetzen des Bausteinnamens durch den Textinhalt des Bausteines) erforderlich macht oder der Bausteinname wird mit dem Kommando "AUSF" abgeschlossen.

Das bewirkt, daß sofort der Bausteintext eingesetzt wird. Für die nachträgliche Bausteinsubstitution gibt es die Funktion „BAUSTEINSUBSTITUTION", mit der alle in einem Text enthaltenen Bausteine eingefügt werden.

Funktionskomplex Dateibehandlung

In TEXT40 laufen eine Reihe von Vorgängen mit Dateien automatisch ab. Das bedeutet, daß der Nutzer diese Vorgänge nicht gezielt beeinflussen kann und seine Mitwirkung nicht erforderlich ist.

Darüber hinaus werden zur Manipulation mit Dateien drei Funktionen zur Verfügung gestellt. Der Nutzer hat die Möglichkeit, Dateien

- umzubenennen,
- zu kopieren oder
- zu löschen.

Funktionskomplex
Textübernahme/-übergabe

Der Programmkomplex „Textübernahme/-übergabe" ist für die Bearbeitung von Dateien vorgesehen, die

- mit anderen Systemen erstellt wurden,
- an andere Systeme übergeben werden sollen und
- für die Druckausgabe vorgesehen sind.

Die Übernahme von Texten aus oder von anderen Textverarbeitungssystemen erfolgt derart, daß in diesen Texten enthaltene Steuerzeichen entfernt werden und der „reine" Text weiterverarbeitet wird. Analog erfolgt die Ausgabe von TEXT-40-Texten an andere Systeme.

Funktionskomplex
Serienbriefverarbeitung

Mit Hilfe des Programmkomplexes „Serienbriefverabeitung" können gleichlautende Schreiben, die sich nur durch die Anschrift unterscheiden, für den Druck aufbereitet bzw. gedruckt werden. Dabei gehören zur Anschrift:

- die Adresse,
- die Anrede und
- die Grußformel.

Bestimmte Anschriftengruppen können durch den Nutzer als solche zusammengefaßt und unter einem Anschriftengruppennamen abgelegt und bei Bedarf aufgerufen werden (Verteilerschlüssel). Die Serienbriefverarbeitung beinhaltet:

- Anschrifteneingabe
- Anschriftenänderung
- Anschriften löschen
- Anschriftenliste drucken
- Auftrag zusammenstellen
- Auftrag ausführen.

Funktionskomplex Terminüberwachung

Die Terminüberwachung des TEXT 40 dient dazu, sich jederzeit einen Überblick über anstehende Termine, zu erledigende Arbeiten u. a. informieren zu können. Dazu ist es möglich, Termine einzugeben, zu ändern oder zu löschen. Diese Termine können abgefragt werden, oder man kann sich akustisch auf einen Termin hinweisen lassen.

Zu einem Termin gehören folgende Informationen:

- Datum und eventuell Uhrzeit,
- Kurzbezeichnung des Ereignisses und
- Bearbeitungszeitraum.

- Termine abfragen

Wird diese Funktion angewählt, erscheint auf dem Bildschirm das aktuelle Datum und eine Liste aller der Termine, deren Datum bereits überfällig ist, wo das Datum mit dem aktuellen Datum übereinstimmt bzw. wo das Datum im Bereich „aktuelles Datum + Bearbeitungszeitraum" liegt. Beispielsweise erscheint ab 16.10. der Hinweis auf einen Bericht, der am 30.11. fällig ist und für

dessen Bearbeitung eine erforderliche Zeitspanne von 14 Tagen angegeben wurde.
Mit den Cursorpositionierfunktionen kann die Terminliste nach oben/unten gerollt werden. Es besteht die Möglichkeit, das angezeigte Datum (normalerweise das aktuelle) zu überschreiben (d.h. zu ändern) und somit die Termine für einen beliebigen Tag abzufragen.

Funktionskomplex Formate

Das Textverarbeitungssystem TEXT 40 gestattet dem Nutzer, Texte ohne Beachtung des gewünschten zu druckenden Formates fortlaufend einzugeben. Während der Eingabe wird der Text automatisch formatiert. Dazu ist es erforderlich, daß bestimmte wiederkehrende Formate in einer TEXT-40-internen Datei gespeichert und somit alle erforderlichen Informationen zu einem gewünschten Format über eine Formatbezeichnung verfügbar sind.
Zusätzlich zu einer Reihe von standardisierten, in TEXT40 vorhandenen Formaten kann sich jeder Nutzer für spezielle Zwecke eigene Formate definieren, diese beliebig ändern oder wieder löschen.

- Formatdefinitionen

Nach der Auswahl dieser Funktion erscheint eine Bedienerführung, die verlangt, daß ein Name, unter dem dieses Format abgelegt werden soll, eingegeben wird. Auf dem Bildschirm erscheinen alle in einer Formatdefinition angebbaren Parameter, initialisiert mit den Werten des Formates A4. Mit dem Cursor können alle Parameter angewählt und bei Bedarf geändert werden.

Folgende Parameter gehören zum Format:

– Formatname
– Gesamtzeilenzahl
Anzahl der möglichen Druckzeilen (Ganzzeilen auf dem Blatt)
– erste Schreibzeile
Abstand des Schriftfeldes vom oberen Blattrand in Ganzzeilen
– letzte Schreibzeile
Abstand der letzten Schriftzeile vom oberen Blattrand in Ganzzeilen
– Zeilenlänge
max. Anzahl von Schriftzeichen pro Zeile
– Zeilenabstand
Angabe in Halbzeilen
– linker Rand
Anzahl der Leerzeichen vor der ersten Schreibposition bezogen auf den linken Blattrand
– Pos. der Seiten-Nr. oben
Position der Einerstelle der Seitennummer von der linken oberen Blattecke in Ganzzeilen/Zeichen
– Pos. der Seiten-Nr.
wie oben
– Tabulatoren
Position vom linken Schriftfeldrand

- Formatkorrektur

Mit dieser Funktion können alle selbst definierten Formate beliebig verändert werden. Nach Eingabe des Formatnamens können die erscheinenden Parameter angewählt und überschrieben werden. Der Abschluß der Funktion „Formatkorrektur" erfolgt mit dem Kommando "AUSF".
Die Korrektur der Standardformate ist aus Sicherheitsgründen nur mit Kenntnis bestimmter Schlüsselworte möglich.

Funktionskomplex Standards

Das Textverarbeitungssystem TEXT 40 ist so aufgebaut, daß dem Nutzer möglichst viele monotone, sich häufig wiederholende Arbeiten abgenommen werden können.
So enthalten Parameter, die bei bestimmten Funktionen abgefragt werden, Standardwerte, so daß diese Parameter nur dann einzugeben sind, wenn sie vom Standard abweichen.
Entpricht ein Standard-Parameter nicht den Vorstellungen eines Nutzers, so kann er diese nach seinen Erfordernissen abändern. Eine komplette Übersicht über die Standardparameter erhält man über die Bedienerführung „STANDARDS".

aspekte vorschau

2/87
Unter dem Titel
CAD/CAM
Grundlagen – Anwendungen – Erfahrungen
werden in edv-aspekte 2/1987 Beiträge der im Dezember 1986 in Berlin stattgefundenen CAD/CAM-Konferenz der Wissenschaftlich-Technischen Gesellschaft für Meß- und Automatisierungstechnik (WGMA) der Kammer der Technik veröffentlicht. Ausgewählt für diese Publikation wurden Beiträge zu den Themenkomplexen

- Theoretische und methodische Grundlagen
- Geräte- und programmtechnische Basis für die rechnergestützte Arbeit
- Anwendungen
- CAD in der Automatisierungstechnik
- CAM-flexible Automatisierung
- CAD/CAM in der Elektrotechnik/Elektronik
- CAD/CAM im Bauwesen

Mit diesen Materialien werden sowohl Erfahrungen aus der Wissenschaft wie auch fortgeschrittener Anwender in den Kombinaten dargestellt. edv-aspekte 2/87 gibt bei der Schaffung von weiteren CAD/CAM-Arbeitsstationen in unserer Volkswirtschaft Anregungen und Anleitung.

Bestellungen sind zu richten an den Verlag Die Wirtschaft, Abteilung Vertrieb, Am Friedrichshain 22, Berlin, 1055.

Ein Hinweis

Im aspekte-Heft 4/86, Seite 43, fehlten auf Grund eines technischen Fehlers zum Artikel „Rationelle Programmierung dialogintensiver Programme" die Autorenangaben. Die Autoren sind: Berger; Zickert, VEB Elektro-Anlagenbau Zwickau. Wir bitten um Verständnis.

Die Redaktion

Softwarebausteine für den AC 7100

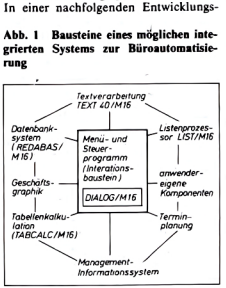
Wilfried Heymel
VEB Robotron-Projekt Dresden

Softwarebausteine sind vorgefertigte oder bereits schlüsselfertige Softwarekomponenten, die vom Nutzer ohne oder nur mit geringem Anpassungsaufwand zur Lösung der eigenen betriebsorganisatorischen Probleme eingesetzt werden können. Sie stellen zusammen mit den entwickelten Programmen den Fundus von Anwendungsprogrammen dar, mit denen der Anwender als EDV-Endnutzer arbeitet.
Physisch sind Softwarebausteine linkbare oder direkt aufrufbare Programme oder Programmsysteme.
Ziel ist es, etappenweise ein integriertes System von untereinander paßfähigen Softwarelösungen zu schaffen, die sowohl als Einzelkomponenten als auch im Verbund nutzbar sind. In der ersten Etappe erfolgt das Schaffen von funktionell aufeinander abgestimmten Einzellösungen.
Zunächst werden folgende Produkte angeboten:

- Dialogbaustein-DIALOG/M 16
- Listenprozessor LIST/M 16
- Tabellenkalkulationsprogramm TABCALC/M 16.

Diese Softwarekomponenten werden ergänzt mittels folgender in Entwicklung befindlicher Programmsysteme:
– Relationales Datenbanksystem REDABAS/M 16
– Textverarbeitungssystem TEXT 40.
In einer nachfolgenden Entwicklungsphase ist die Erhöhung des Integrationsgrades, die funktionelle Erweiterung der vorliegenden Bausteine sowie die Ergänzung durch neue Lösungen geplant.
Abb. 1 verdeutlicht, welche Funktionen ein solches integriertes Bausteinsystem beinhalten könnte.
Die nachfolgend vorgestellten Softwarelösungen sind vorrangig ausgelegt für den Einsatz auf Arbeitsplatzcomputern AC 7100 unter dem Betriebssystem SCP 1700. Sie sind jedoch mit Einschränkungen lauffähig/anpaßbar auf ähnliche Rechnersysteme anderer Hersteller. Mit Hilfe der weitgehend durchgängigen Verwendung der Programmiersprache C ist eine hohe Portabilität und Hardwareunabhängigkeit gegeben. LIST und DIALOG sind auch lauffähig auf robotron K 1630 unter dem Betriebssystem MUTOS.

Abb. 1 Bausteine eines möglichen integrierten Systems zur Büroautomatisierung



Listenprozessor LIST/M 16

Charakteristik und Aufgabenstellung des Programmes

LIST/M 16 ist ein flexibles Listengenerierungssystem. Es dient sowohl EDV-Fachleuten als auch EDV-unkundigen Anwendern zur kurzfristigen Darstellung ihrer in Dateien gespeicherten Daten entsprechend dem gewünschten Listenaufbau.
Es ist einfach zu benutzen und umfassend in seinen Möglichkeiten. Mit Hilfe einer einfachen Anweisungssprache ist es möglich, die auszugebenden Daten zu charakterisieren, Datenmanipulation festzulegen und das gewünschte Listenbild zu beschreiben, ohne Programmier- oder Betriebssystemkenntnisse zu besitzen.
LIST/M 16 ist programmiersprachenunabhängig einsetzbar, d. h., die Programme, mit denen die durch LIST zu verarbeitenden Dateien erzeugt wurden, können in einer beliebigen, im SCP 1700 unterstützten Programmiersprache geschrieben sein. Unter MUTOS 1630 ist LIST allerdings nur im Zusammenhang mit C-Programmen einsetzbar.
Mittels Abspeicherung der in einen Zwischencode übersetzten Listenbeschreibung lassen sich die Aufbereitung und Ausgabe einer Liste jederzeit wiederholen.
Relativ aufwendige Listendefinitionen, die zudem routinemäßig abgearbeitet werden sollen, lassen sich durch LIST/M leicht fixieren und zu beliebigen Zeitpunkten abarbeiten. LIST/M 16 ist an keine spezielle Hardware gebunden. Die minimalen Hardware-Anforderungen sind folgende:
– Zentraleinheit mit mindestens 128 kByte Hauptspeicher
– Externspeicher (Platte oder Diskette)
– Bildschirm
– Tastatur
– Drucker.
Das Programmsystem ist so konzipiert, daß Hardware-Schnittstellen (Bildschirm, Drucker) sich in speziellen Moduln befinden, die gegebenenfalls leicht an abweichbare Hardware-Bedingungen angepaßt werden können.

Arbeitsweise und Funktionen von LIST/M 16

Die Arbeitsweise des Listengenerators wird durch die Auswertung der in Sprachanweisungen formulierten Listenbeschreibungen und der im Aufrufkommando enthaltenen Spezifikationen gesteuert. Die Listenbeschreibungsquelle enthält Aussagen zu Inhalt und Aufbau der vom Anwender gewünschten Liste.
Mit Hilfe des Aufrufkommandos spezifiziert der Anwender direkt oder im Dialog Angaben zur Arbeitsweise (in Form von Optionen), zum Ablauf von LIST/M sowie zur Verwendung nicht standardgemäßer E/A-Geräte.
Der Listengenerator durchläuft folgende Arbeitsphasen (Abb. 2):
– Syntaktische und semantische Analyse
Die Sprachanweisungen der Listenbeschreibungsquelle werden auf syntaktische und semantische Richtigkeit geprüft; ein compilierter Code (Zwischencode) wird erzeugt, der auch als Datei abgelegt werden kann
– Erstellen und Sortieren des Hitfiles
Das Hitfiles enthält entsprechend der Listenbeschreibungsquelle die für den Druck und die Listenaufbereitung selektierten Daten, die in der, durch Li-

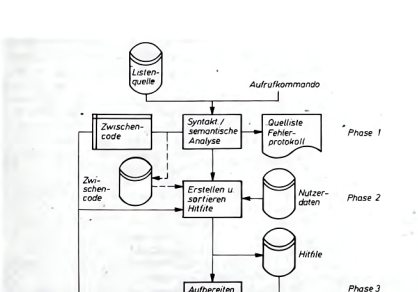


Abb. 2 Wirkungsprinzip des Listengenerators

stenbeschreibung vorgegebenen Reihenfolge sortiert werden

– Aufbereiten und Ausgabe der Listen

Eine oder mehrere Listen werden entsprechend dem gewünschten Listenbild für den Druck aufbereitet und ausgegeben; dabei können listenspezifische Datenmanipulationen durchgeführt und deren Ergebnisse im Listenbild sichtbar gemacht werden.

Für den Ablauf von LIST/M gibt es zwei Möglichkeiten:

- LIST/M wird die Listenbeschreibungsquelle als Eingangsdatei zugewiesen. Alle Phasen des Programmsystems laufen in der beschriebenen Weise ab
- LIST/M wird der bereits zu einem früheren Zeitpunkt compilierte und als Datei abgelegte Code der Listenbeschreibungsquelle (Zwischencode) angeboten. In diesem Fall wird an Stelle der ersten Arbeitsphase (syntaktische und semantische Analyse) das Einlesen der Zwischencodedatei in den Hauptspeicher vorgenommen.

Normalerweise werden alle Phasen des Listengenerators während der Abarbeitung einer Listenbeschreibungsquelle nur einmal durchlaufen (auch wenn mehrere Listen generiert werden sollen). Aus Gründen der Speicherkapazität ist es auch möglich, die letzten Arbeitsphasen für jede zu erstellende Liste gesondert zu aktivieren. Dazu werden diese Phasen zyklisch, entsprechend der Anzahl der zu erzeugenden Listen aufgerufen.

Mit der Anwendung von LIST/M 16 werden folgende allgemeine Funktionen verwirklicht:

- automatische Zeilen- und Seitenzählung
- automatischer Seitenwechsel mit Druck von Datum, Seitennummer und einer spezifizierbaren Seiten- und Spaltenkopfzeile
- wahlweise Speicherung des Zwischencodes der Listenbeschreibungsquelle als Datei für eine spätere Ausführung der Listenerzeugung
- wahlweise Ausgabe der Quell- und Fehlerliste (Syntaxfehler) der Listenbeschreibung
- Erzeugung mehrerer verschieden aufgebauter Listen mittels einer einzigen Listenbeschreibungsquelle
- Einzel- oder Gesamtreporterarbeitung der Listen einer Listenbeschreibungsquelle
- Ausgabe mehrerer Nutzerdateien nacheinander in der Listenbeschreibung angegebenen Form mit Hilfe einer einmal compilierten Listenbeschreibungsquelle (Die Nutzerdateisätze müssen die gleiche Struktur besitzen!).

Mit Hilfe der LIST-Sprache können folgende Möglichkeiten für die Listenerstellung genutzt werden:

– Spezifikation einer Anwenderidentifikation

Die Anwenderidentifikation ist die Überschrift aller Listen einer Listenbeschreibungsquelle und erscheint auf der ersten Zeile jeder Seite. Sie dient der Identifikation des Anwenders

– Spezifikation der Listenkopfzeile

Die Listenkopfzeile, einschließlich Datum und Seitennummer, erscheint auf jeder Seite einer Liste im Anschluß an die Anwenderidentifikation (falls vorhanden)

– Spezifikation der Spaltenkopfzeile

Wurden in der Listendefinition keine speziellen Spaltenköpfe angegeben, so werden die internen Bezeichnungen der zu druckenden Spalten als Spaltenkopf benutzt

– Spezifikation der Listenfußzeilen

Die Listenfußzeilen bilden den Abschluß einer Liste

– Benutzung von Hauptspeicherfeldern zwecks Datenmanipulation

Hauptspeicherfelder können explizit durch eine LIST/M-Anweisung definiert und mit einem Anfangswert belegt werden. Außerdem ist die automatische Definition von Hauptspeicherfeldern möglich, indem in Datenmanipulationsanweisungen von LIST/M ein neuer Name benutzt und ihm eine Zahl oder der Wert eines arithmetischen Ausdrucks zugewiesen wird. Mit Hauptspeicherfeldern kann manipuliert, danach sortiert und sie können ausgegeben werden

– Ausführen arithmetischer Berechnungen

Nutzer- und Hauptspeicherdaten kön-

nen für arithmetische Berechnungen (vier Grundrechenarten und ganzzahlige Division, die als Ergebnis den Rest liefert) verwendet werden. Diese Berechnungen können sowohl in den Detail- als auch ausschließlich in den Zwischensummenzeilen einer Liste vorgenommen werden
– Dekodieren verschlüsselt abgespeicherter Daten
Durch Angabe einer Dekodiertabelle können verschlüsselt abgespeicherte Daten in der Liste in ihrem Klartext erscheinen. Ebenso kann nach diesen dekodierten Werten selektiert und sortiert werden
– Definition von Kontrollunterbrechungen
Zum Druck von Zwischen- und Gesamtsummenzeilen ist es erforderlich, Kontrollunterbrechungen festzulegen. Eine Kontrollunterbrechung veranlaßt den Druck von Zwischensummen immer dann, wenn gekennzeichnete Datenfelder ihren Wert ändern. Das Summieren wird von LIST/M automatisch vorgenommen. Außerdem ist es durch eine spezielle LIST/M-Anweisung möglich, arithmetische Berechnungen nur nach einer Kontrollunterbrechung ausführen und drucken zu lassen
– Selektieren der Sätze der Nutzerdatei
Durch Angabe von logischen Bedingungen (<, ≤, >, ≥, =, !=, NOT, AND, OR) sind Sätze aus der Nutzerdatei für den Listendruck auswählbar. Diese Auswahlkriterien können zusätzlich mit einem Buchstaben, dem sogenannten „TAG"-Wert, gekennzeichnet werden. Erfüllt ein Satz der Nutzerdatei eine logische Bedingung, bekommt er den der Bedingung entsprechenden TAG-Wert zugeordnet. Für jeden zu druckenden Satz ist intern der Platz für einen TAG-Wert vorgesehen. Der TAG-Wert kann für den bedingten Druck eines Datenfeldes (Druck nur, falls zugehöriger TAG-Wert ein bestimmter Buchstabe ist), für den Druck des TAG-Wertes selbst und als Sortierfeld verwendet werden.
– Auf- und absteigendes Sortieren der für die Druckausgabe vorgesehenen Sätze der Nutzerdatei
Es kann numerisch und alphanumerisch nach ein oder mehreren Sortierfeldern sortiert werden. Zusätzlich kann nach dem TAG-Wert und nach Hauptspeicher- und Datenfeldern sortiert werden, die nicht explizit in der Liste erscheinen sollen
– Unterdrückung der Ausgabe einzelner Daten in den Detailzeilen einer Liste
Listen werden zum Teil übersichtlicher und leichter lesbar, wenn einige Daten (z. B. nach denen sortiert wird) nur in den Zwischensummenzeilen erscheinen. Die Ausgabe dieser Daten kann in den Detailzeilen unterdrückt werden.

Überblick über die Sprachanweisungen

Die Listenbeschreibungssprache besteht aus 14 Sprachanweisungen, die in fünf Komplexe unterteilt werden können, und von denen der weitaus größte Teil wahlfrei zu benutzen ist (in der nachfolgenden Übersicht durch (w) gekennzeichnet).

① Steueranweisungen
OPTION-Anweisung (w)
– Festlegen des Formats der Drucklisten
– Verfügbarkeit von Kleinbuchstaben auf dem Ausgabegerät
USER-Anweisung (w)
– Spezifikation der Anwenderidentifikation

② Anweisungen zur Datei- und Datendefinition
FILE-Anweisung
– Beschreibung der auszuwertenden Nutzerdatei durch Name, Speichermedium und Satzlänge
DATA-Anweisung
Es brauchen nur die Daten der Nutzerdatei beschrieben werden, die für Berechnungen, Selektion und Sortieren von Sätzen oder den Druck der Listen benötigt werden
– Beschreibung eines Datenfeldes der Nutzerdatei durch symbolischen Namen, Position im Nutzersatz und wahlfreie Angaben über internes Format, Ausgabeposition, externes Format in der Liste und Spaltenkopfzeile
DEFINE-Anweisung (w)
– explizite Definition von Hauptspeicherfeldern durch symbolischen Namen und wahlfreie Angaben bezüglich Anfangswert, internes Format, Ausgangsposition, externes Format in der Liste und Spaltenkopfzeile

③ Anweisungen zur Datenmanipulation
SET-Anweisung (w)
– Berechnung eines arithmetischen Ausdrucks vor Beginn des Selektier- und Sortierprozesses
– Automatische Definition eines Hauptspeicherfeldes
DECODE-Anweisung (w)
– Dekodierung eines verschlüsselt abgespeicherten Datenfeldes der Nutzerdatei mit Hilfe einer angegebenen Dekodierungstabelle
– automatische Definition eines Hauptspeicherfeldes, das die dekodierten Werte enthält

④ Anweisungen zur Listenspezifikation
REPORT-Anweisung
Jede Listenspezifikation beginnt mit der REPORT-Anweisung
– Spezifikation von Listenkopfzeilen
– Festlegen eines speziellen Listenformats
SELECT-Anweisung (w)
– Selektieren von Sätzen der Nutzerdatei nach vorgegebenen Auswahlkriterien (logische Bedingungen und deren Verknüpfungen)
– Zuordnen von möglichen „TAG"-Werten
CONTROL-Anweisung (w)
– Festlegen der Felder, nach denen auf- oder absteigend sortiert werden soll
– Definition der Kontrollunterbrechungen
– Zwischensummenbeschreibung (Text, der vor den zu druckenden Zwischensummen erscheint)
– Vorschubsteuerung nach Kontrollunterbrechungen
CSET-Anweisung (w)
– Berechnung eines arithmetischen Ausdrucks nach dem Selektier- und Sortierprozeß zum Zeitpunkt einer Kontrollunterbrechung
– Automatische Definition eines Hauptspeicherfeldes
PRINT-Anweisung
– Festlegen der pro Zeile zu druckenden Nutzerdaten, Hauptspeicherfelder und eventuell des „TAG"-Wertes durch die symbolischen Namen und wahlfreie

Angabe der Ausgabepositionen und der externen Formate
– Festlegen der automatisch zu summierenden Felder
– Auswerten eines „TAG"-Wertes für den bedingten Druck eines Feldes in den Detailzeilen einer Liste
FOOT-Anweisung (w)
– Spezifikation von Listenfußzeilen
Ⓢ Ende der Listenbeschreibung
END-Anweisung
– Abschluß des Listenjobs

DIALOG/M 16

Charakteristik des Programmes

DIALOG/M 16 ist ein programmtechnisches Werkzeug zur Realisierung eines Dialoges mit dem Bediener und ist als solches aus beliebigen Anwendungsprogrammen aufrufbar. Dazu werden feststehende (nicht rollende) aus Variablen und Konstanten bestehende Masken auf den Bildschirm projiziert, deren Variablen entweder vom Programm mit Werten oder vom Bediener über die Tastatur mit Eingabedaten gefüllt werden. Die Bildschirmmaske ist vom Benutzer mittels einfach zu handhabender Sprachmittel zu beschreiben. Durch die Beschreibung wird natürlich auch die Größe der Maske auf dem Bildschirm festgelegt. Der durch die Maske nicht belegte Teil des Bildschirmes steht als Rollbereich zur Verfügung.
Zur Ausführungszeit wird aus dem Nutzerprogramm ein Laufzeitsystem aufgerufen, welches den gewünschten Dialog realisiert. Durch diese CALL-Schnittstelle ist DIALOG/M weitestgehend sprachunabhängig nutzbar.
Durch die Nutzung von DIALOG/M 16 wird dem Anwender die Gestaltung und Steuerung von Bildschirmdialogen in seinen Programmen wesentlich erleichtert. Ohne spezielle Kenntnisse der Terminal-E/A-Funktionen können aus beliebigen Anwendungsprogrammen unter voller Ausnutzung der gegebenen Hardwarefunktion Dialog- und -ausgaben realisiert werden.

Arbeitsweise des Dialogwerkzeuges

In Abb.3 ist der prinzipielle Aufbau des Dialogwerkzeuges und das Zusammenspiel der einzelnen Komponenten dargestellt. Das System besteht aus drei getrennten Komponenten:
– dem Maskengenerator
Er analysiert die vom Nutzer erzeugte Maskenbeschreibung, prüft sie auf syntaktische und semantische Richtigkeit und übersetzt sie in einen internen Mikrocode.
In der Maskenbeschreibung festgestellte Fehler werden dem Bediener mitgeteilt, wobei zwischen Warnungen und schweren Fehlern unterschieden wird. Bei letzteren ist eine Weiterarbeit mit der unkorrigierten Maskenbeschreibung nicht möglich
– dem Dialogtester
Er veranlaßt die Simulation der Maske auf dem Bildschirm auf der Grundlage der internen Maskenbeschreibung, ohne daß ein rufendes Nutzerprogramm benötigt wird. Damit ist eine visuelle Kontrolle der Maske ohne Einbindung in ein Nutzerprogramm möglich.
Er ist also ein Hilfsmittel zur Prüfung der Maskenbeschreibung, ob damit das gewünschte Bild erzielt wird
– dem Dialoghandler
Er stellt eine Funktionshierarchie dar,

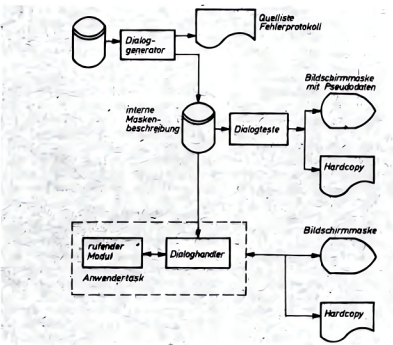


Abb. 3 Wirkungsprinzip des Dialogbausteines

die mit dem Nutzerprogramm zu einem lauffähigen Programm zusammengebunden und von diesem aufgerufen wird. Er realisiert die in der Maskenbeschreibung festgelegten Terminal-Ein- und Ausgaben. Der Datenaustausch mit dem Nutzerprogramm erfolgt über Parameterlisten.
Der Dialoghandler arbeitet so, daß entsprechend der internen Maskenbeschreibung eine Bildschirmmaske ausgegeben wird, in der alle Konstanten und Ausgabevariablen enthalten sind. Dies geschieht bildschirmseitengerecht, d. h., ist eine Bildschirmseite voll, wird in den Eingabemodus umgeschaltet, und alle Eingabevariablen dieser Seite werden vom Bediener abgefordert. Dabei läuft der Kursor automatisch alle Eingabevariablen von links oben nach rechts unten ab.
Erstreckt sich eine Maske über mehrere Seiten, wird nach Entgegennahme der letzten Eingabevariablen wieder in den

Ausgabemodus zurückgeschaltet und die nächste Seite auf den Bildschirm projiziert. Nach Behandlung der letzten Seite erfolgt Rückgabe der Steuerung an das rufende Programm. Alle Eingabevariablen stehen an den entsprechenden Adressen dem Nutzerprogramm zur Verfügung. Für alle Variablen erfolgt entsprechend ihrem Typ eine automatische Konvertierung.

Aufbau einer Maskenbeschreibung

Eine Maskenbeschreibung wird als bildliche Darstellung der gewünschten Maske (Bildschirm-LAYOUT) definiert. Zur Kennzeichnung der Beschreibung und zur Festlegung von Attributen stehen Sprachanweisungen zur Verfügung. Eine Bildschirmmaske ist nach folgendem Schema anzulegen:

- LAYOUT ... Einleitende Anweisung enthält Angaben über Eigenschaften des zu erzeugenden Bildes und zur Interpretation der bildlichen Darstellung
- bildliche Darstellung
- Konstanten werden durch ihren vollen Text,
- Variable durch spezielle Zeichen dargestellt
- VARIABLES ... Attributbeschreibung für Variable
- WINDOW ... Durch diese Anweisung kann die Maske in einzelne Windows unterteilt werden, die wie eine selbständige Maske aufgerufen werden können
- CONSTANTS ... Attributbeschreibung für Konstanten. Diese Anweisung ist nur erforderlich, wenn vom Standard abweichende Attribute zugewiesen werden sollen.

Mittels Attributanweisungen kann unabhängig von den Funktionen des benutzten Terminals die Darstellungsform der Variablen und Konstanten auf dem Bildschirm festgelegt werden. Abhängig von den vorhandenen Hardwarefunktionen sind folgende Attribute möglich:

– intensiv hell/normal hell
– akustisches Signal
– rechts/links Justierung
– Prüfung der Ausgabewerte auf Gültigkeit
– Speichern von Werten zur wiederholten Verwendung
– Schätzen von Bildschirmbereichen vor Überschreibungen.

Entsprechend dem internen Datentyp der Variablen im Nutzerprogramm wird eine automatische Konvertierung in die gewünschte externe Darstellung durchgeführt.

Eine Bildschirmmaske kann aus beliebig vielen Bildschirmseiten bestehen und in beliebig viele Abschnitte (Windows) unterteilt sein, wobei sich die Windows gegenseitig überlappen dürfen.

Windows sind in horizontaler und vertikaler Richtung möglich. Jedes Window kann wie eine separate Maske genutzt werden.

Führung des Dialoges

Zur Erhöhung des Bedienkomforts wurden verschiedene Tasten als Funktionstasten in die Steuerung des Dialoghandlers einbezogen. Folgende Funktionen stehen damit zur Verfügung:

– Datenkorrektur

Sofort erkannte Eingabefehler in der aktuellen Variablen können auf die übliche Weise durch zeichenweises Rückführen des Kursors korrigiert werden. Darüber hinaus kann durch Druck der Funktionstaste „Datenkorrektur" die aktuelle Seite bzw. das aktuelle Window neu angegeben werden. Alle bereits vorher eingegebenen Werte bleiben dabei erhalten. Der Kursor ist wieder auf die erste Eingabevariable positioniert, so daß von da aus alle Variablen entweder korrigiert oder mit ihrem bisherigen Wert bestätigt werden können

– Übernahme des Standardwertes

Damit ist es möglich, Standardwerte für einzelne Variable oder für die gesamte Seite bzw. für das aktuelle Window als aktuelle Eingabe zu bestätigen. Als Standardwerte können entweder bei der Maskenbeschreibung festgelegte Initialisierungswerte oder beim vorhergegangenen Aufruf getätigte und zu diesem Zweck gesicherte Eingabevariable gelten

– Abbruch der Dialogarbeit mit oder ohne Wiederanlauf

Die Arbeit des Handlers wird sofort beendet und ein entsprechender Rückkehrcode an das rufende Programm gemeldet

– Seitenwechsel

Der Nutzer kann damit in den Bildschirmseiten blättern, ohne durch Eingaben die Seiten abzuschließen.

– Druck der aktuellen Bildschirmseite

Fehlerbehandlung und Nachrichten an den Bediener

In der Maskenbeschreibung wird festgelegt, welcher Bereich des Bildschirmes als Maskenbereich belegt wird. Die auf den Maskenbereich folgende Zeile dient zur Nachrichtenübermittlung an den Bediener. Hier werden in intensiver Helligkeit Fehlerzustände an den Bediener gemeldet. Die Nachricht bleibt solange sichtbar und das Programm im Wartezustand, bis der Bediener die Nachricht mittels Druck einer Taste quittiert hat. Die Nachrichtenzeile kann darüber hinaus zur Übermittlung von Informationen aus dem Nutzerprogramm verwendet werden.

Tabellenkalkulationsprogramm TABCALC/M16

Charakteristik des Programmes

Annähernd gleichlaufend mit dem Produktionsbeginn der neuen 16-Bit-Arbeitsplatzcomputer A 7100 wird für diese Rechner unter dem Betriebssystem SCP 1700 auch ein Tabellenkalkulationsprogramm bereitgestellt; in seinen Grundfunktionen ähnlich dem KP, welches bereits seit einiger Zeit für die 8-Bit-Bürocomputer angeboten wird.

Die Grundideologie der Tabellenkalkulation besteht darin, daß im Hauptspeicher in Form einer Matrix ein sogenanntes Arbeitsblatt (Worksheet) simuliert wird. Jede Zelle auch Felder einer Matrix ist über ihre Koordinaten eindeutig adressierbar. Bei TABCALC sind gleichzeitig mehrere Arbeitsblätter im Hauptspeicher möglich.

Die einzelnen Felder einer Matrix können mittels

- Textketten,
- numerischer Werte oder
- Berechnungsformeln

gefüllt werden. Textketten und Berechnungsformeln dürfen maximal 116 Zeichen lang sein. Als Operanden in Berechnungsformeln können Absolutwerte oder Adressen von Feldern verwendet werden, die ebenfalls numerische Werte oder Berechnungsformeln enthalten können.

Auf diese Weise ist die Verkettung von Formelfeldern in beliebigen Tiefen möglich. Jedes Formelfeld besitzt zwei externe Darstellungsformate:
– den Formelmodus
Es wird die gespeicherte Formel selbst sichtbar gemacht
– den Ergebnismodus
Im Ergebnismodus liegt mit der Eingabe einer Formel sofort das Ergebnis vor. Bei verketteten Formeln werden bei Änderung einer Bezugsgröße die Ergebnisse aller Formeln, in denen die Bezugsgröße direkt oder indirekt Verwendung findet, sofort automatisch korrigiert. Dieser Automatismus kann jedoch auch blockiert werden. Das ist dann sinnvoll, wenn erst nach umfangreicheren Formeleingaben bzw. -änderungen eine Neuberechnung erwünscht ist.
Zur Sichtbarmachung des Tabelleninhaltes kann der Bildschirm durch Positioniertasten und -anweisungen als *Fenster* über jeden beliebigen Bereich der Tabelle gefahren werden. Weiterhin ist die Ausgabe des Tabelleninhaltes in aufbereiteter Form direkt über einen Drucker oder als druckbare Textdatei auf eine Diskette möglich.
Eingerichtete Tabellen können jederzeit komplett oder teilweise auf eine Diskette gesichert und von dort auch wieder aktiviert werden.
Alle Funktionen des TABCALC werden dem Benutzer mit einem Kommandomenü geboten. In jeder beliebigen Phase können detaillierte Helpinformationen abgefordert werden.

Arbeitsweise und Funktionen

TABCALC meldet sich nach dem Start mit einem Koordinierungssystem auf dem Bildschirm. Damit wird ein Ausschnitt einer Tabelle (Arbeitsblatt) dargestellt, die
– horizontal in 63 Spalten untergliedert ist, wobei die Spalten 1 bis 26 mit den laufenden Buchstaben A bis Z und die nächsten 37 mit den Doppelbuchstaben AA bis BK bezeichnet werden und
– vertikal aus 255 Zeilen besteht, die von 1 bis 255 numeriert sind.
Bei der Arbeit mit TABCALC dienen diese Koordinaten als Adressen der einzelnen Felder der Tabelle. Es bezeichnet beispielsweise A1 das Feld in der linken oberen und BK255 das Feld in der rechten unteren Ecke der Tabelle.
Neben den einzelnen Feldern der Tabelle kann mit Strukturelementen wie
– Zeile
– Spalte
– Reihe, das heißt benachbarte Felder, die einen Teil einer Zeile oder Spalte bilden
– Blöcke, das sind Felder, die in einem rechteckigen Tabellenausschnitt liegen und durch die Koordinaten der linken oberen und der rechten unteren Eckfelder begrenzt sind
gearbeitet werden.
Jede Zelle der Matrix ist im Standard 9 Byte lang (externe Darstellungsform). Damit ist ein Tabellenbereich von 19 Zeilen × 8 Spalten (bei 24zeiligem Bildschirm) auf dem Bildschirm darstellbar.
Über Steuertasten rechts/links/hoch/tief kann der Tabellenkursor auf jede beliebige Zelle des Bildschirmausschnittes positioniert werden. Beim *Hinausfahren* des Cursors aus dem Bild wird der Bildschirmausschnitt automatisch verschoben.
Ferner kann über Sprungbefehle jede beliebige Zelle adressiert werden. Liegt das Sprungziel außerhalb des Bildschirmfensters, wird die betreffende Zelle automatisch zum linken oberen Eckfeld des neuen Bildschirmausschnittes.
Die auf den Tabellenausschnitt folgenden Zeilen werden zur Anzeige des Tabellenstatus, des Inhaltes der aktuellen Felder, zur Übermittlung von Nachrichten sowie als Tabelleneingabezeile benutzt. Hier erfolgt im Kommandostatus auch die Anzeige des Kommandomenüs.
Über Formatierungskommandos kann das Tabellenformat abweichend vom Standard in vielfältiger Weise geändert werden, wie z. B.
– Spaltenbreite
– Rechts-/Linksbündigkeit von Textketten und numerischen Größen
– Darstellungsformat von numerischen Größen (Integerform, Gleitkommadarstellung, Festkommaform).
Die Formatierung der Tabelle ist mit steigender Priorität möglich für
- die gesamte Tabelle
- eine Spalte
- eine Zeile oder eines Blocks.

Ebenfalls werden über Kommandos weitere Funktionen zur Gestaltung des Arbeitsblattes bereitgestellt, wie z. B.
– Schützen von Tabellenbereichen gegen Überschreiben und Löschen,
– Definition von Titelzeilen,
(als Überschrift gekennzeichnete Zeilen sind nicht in das Rollen des Bildschirmes einbezogen) und
– Teilen des Arbeitsblattes in voneinander unabhängige Bereiche (Windows).

Aufruf von Kommandos – Eingabe von Daten und Formeln

Nach Einschalten des Kommandomodus durch Druck einer speziellen Taste wird im Dialogteil des Bildschirmes das Kommandomenü in Form von Schlüsselwörtern angezeigt. Daraufhin kann das gewünschte Kommando durch Eingabe des oder der ersten signifikanten Zeichen oder durch Positionierung des Kommandokursors ausgewählt werden.
In der Regel bieten die Kommandos weitere hierarchisch gestufte Untermenüs bis zur Auswahl der endgültigen Funktion an. Durch Druck einer Help-Taste können erklärende Hinweise auf dem Bildschirm abgefordert werden. Geübte Anwender können jedoch auch ein komplettes Kommando mit sämtlichen Parametern eingeben, ohne erst die Anforderungszeile des jeweiligen Untermenüs abzuwarten.
Alle Eingaben erfolgen zunächst auch in die Eingabezeile. Von dort werden sie nach Abschluß der Eingabe in das aktive Feld mit dem augenblicklichen Stand des Tabellenkursors gekennzeichnet. Vor jeder Eingabeaktion ist also der Tabellenkursor auf das Feld, in das die Eingabe erfolgen soll, zu positionieren.
Es können nur in ungeschützte Felder Eintragungen vorgenommen werden. Dabei wird generell zwischen Text-, Zahlen- und Formeleingaben unterschieden. Diese verschiedenen Typen werden anhand des ersten in die Eingabezeile eingegebenen Zeichens erkannt und entsprechend behandelt.
Textfelder und Formelfelder können bis zu 116 Zeichen aufnehmen. Numerische Größen dürfen aus maximal 16 Ziffern und einem Vorzeichen bestehen. In Tex-

ten dürfen sämtliche Buchstaben, Ziffern und Sonderzeichen verwendet werden.
Die Kodierung von Formeln erfolgt in der mathematisch üblichen Schreibweise.
Über entsprechende Operationen und Schlüsselwörter können in Berechnungsformeln eine Vielzahl von arithmetischen und logischen Funktionen sowie Kalender-, Finanz- und Spezialfunktionen formuliert und aufgerufen werden.
Zur Änderung von bereits eingegebenen Daten bzw. Formeln steht eine effektive EDIT-Funktion zur Verfügung.

Umstrukturierung von Tabellen

Bei in Arbeit befindlichen Tabellen, die mehr oder weniger bereits mit Daten belegt sind, kann sich die Notwendigkeit der Umstrukturierung nach den verschiedensten Gesichtspunkten ergeben. Über entsprechende Kommandos steht hierfür eine Vielzahl von Funktionen zur Verfügung, z. B. zum
– Löschen von Eintragungen (gesamte Tabelle, Zeilen, Spalten, definierte Bereiche)
– Einfügen und Entfernen von Zeilen bzw. Spalten
– Übertragen von Zeilen bzw. Spalten auf eine andere Position
– Kopieren eines Quell-Tabellenbereiches in einen Zielbereich
– Vervielfältigung von Eintragungen
– Sortieren innerhalb von Zeilen oder Spalten.

Für alle diese Funktionen zur Tabellenumstrukturierung gilt, daß
– rechts bzw. unterhalb der Änderung sich befindende Zeilen oder Spalten in lückenloser Folge neu durchnumeriert und
– in Formeln benutzte Feldreferenzen automatisch korrigiert werden.

Speichern, Laden, Löschen und Ausgeben von Tabellen

Neben den bisher genannten Funktionen zur Arbeit mit den im Hauptspeicher stehenden Tabellen gibt es Funktionen zum Verkehr zwischen Hauptspeicher und externen Speicher bzw. Ausgabegeräten. Es sind:
– Auslagern des Tabelleninhalts auf eine Datei der Diskette

Hierbei kann der Anwender wählen, ob
· die gesamte Tabelle,
· nur die Werte (ohne Formeln) der Tabelle oder
· nur ein Tabellenausschnitt

gespeichert werden soll. Mit diesem Kommando kann ein beliebiger Bearbeitungszustand der Tabelle jederzeit gesichert werden, um im Bedarfsfall wieder zur Verfügung zu stehen.
– Laden von auf der Diskette abgespeicherten Tabellen oder Teile davon in den Hauptspeicher
– Löschen von Dateien auf der Diskette
– Bereitstellen von Tabellen im Druckformat, die auf Drucker, Bildschirm oder Diskette ausgegeben werden können.

Arbeit mit Kommandoprozeduren

Mit den Mitteln des TABCALC selbst können Kommandoprozeduren definiert und abgespeichert werden, in denen alle Kommandos des TABCALC zulässig sind. Die so definierten Arbeitsabläufe sind zu jeder beliebigen Zeit aufrufbar und laufen dann vollautomatisch ab. Kommandoprozeduren können ineinander geschachtelt sein, d.h., daß aus einer Kommandoprozedur weitere Prozeduren aufgerufen werden können. Dadurch ist es möglich, vorgefertigte Arbeitsabläufe aus mehreren Prozedurbausteinen zusammenzusetzen. Die Arbeit mit Kommandoprozeduren stellt ein sehr wesentliches Mittel zur Gestaltung eines effektiven Routinebetriebes dar.

Einsatzgebiete

TABCALC ist überall dort einsetzbar, wo kleinere Datenmengen nach vorgegebenen Algorithmen mit möglichst geringem Aufwand für vorbereitende Algorithmierungs- und Erfassungseinheiten, zu verarbeiten sind. Damit sind die Einsatzmöglichkeiten praktisch unbegrenzt.
Es seien nur einige Beispiele genannt:
– Bilanzierungen
– Kostenplanung und -abrechnung
– Materialanalyse und -planung
– Buchhaltung
– Inventuren
– Kundenkarteien
– Kosten-/Preiskalkulation (z. B. im Gaststättenwesen)
– Projektplanung/-überwachung
– Lohn- und Gehaltsabrechnung.

Ein wesentlicher Vorteil der Tabellenkalkulation besteht darin, daß sämtliche Fixwerte (z. B. Stundenlohn, feste Zuschläge, feste Abzüge in der Lohnrechnung) sowie die notwendigen Berechnungsformeln nur einmal eingegeben werden müssen, jedoch trotzdem jederzeit problemlos fortgeschrieben werden können und sofort zur Abarbeitung bereitstehen.

Software – Was ist das?

Von Prof. Dr. Eberhard Prager und Dr. Evelyn Richter
80 Seiten, Broschur, 5,80 M
Bestellangaben: ISBN 3-349-00025-8
675 958 1/Prager, Software

Der sehr breite Kreis von Lesern, die sich für den wissenschaftlich-technischen, ökonomischen und sozialen Fortschritt interessieren, wird hier über ein neues Element im System der Produktivkräfte und seine Wirkungen informiert. Die ganze Differenziertheit des Inhalts der Software und ihre Funktionen als Arbeitsmittel werden erläutert. Der Produktionsprozeß von Software wird einer politökonomischen Analyse unterzogen, Gedanken zum Gebrauchswert und zum Wert der Software werden ebenso diskutiert wie die Gemeinsamkeiten und Unterschiede zwischen Softwareproduktion, Forschung und Produktion. Die Wege zu höherer Effektivität der Produktion und Nutzung von Software, der gegenwärtige Entwicklungsstand, Probleme und Aufgaben auf diesen Wegen werden erläutert vor allem unter dem Aspekt, wie die Vorzüge des Sozialismus zielstrebig auszuschöpfen sind.
Anforderungen an die Kaderentwicklung werden sichtbar gemacht.

Verlag Die Wirtschaft Berlin
Am Friedrichshain 22
Berlin, 1055

Robotron-Software für mathematische Verfahren

Dieter Wildenhain
VEB Robotron-Projekt Dresden

Die effektive Lösung einer Vielzahl von wissenschaftlichen, technischen und ökonomischen Problemen setzt die Anwendung mathematischer Verfahren bzw. entsprechender Software für die vorhandene Rechentechnik voraus. Die Notwendigkeit hierfür ergibt sich insbesondere aus dem Kompliziertheits- und Verflechtungsgrad dieser Probleme sowie aus dem Umfang der zu verarbeitenden Daten.

Vom Kombinat Robotron wird seit 1969 Software für mathematische Verfahren für EDVA-ESER entwickelt. Derzeit werden die Softwareprodukte STATISTIK-2, STATISTIK-CLUSTER, STAVE-ROOT, OPSI-3, DISKO-2, DISKO-T, SIMDIS-2, EDO-3 und NUMATH-2 zur Lösung von Problemen der Gebiete mathematische Statistik, lineare Optimierung, ganzzahlige Optimierung, Transportoptimierung, diskrete Simulation und numerische Mathematik bereitgestellt. Diese Software ist auf EDVA-ESER mit dem Betriebssystem OS/ES lauffähig.

Software für den Robotron-Mikrorechner K 1630 mit dem Betriebssystem MOOS wird seit 1980 entwickelt. Es stehen die Programmpakete MASTAT 1630 und TRANSPORT 1600 zur Lösung von Problemen der mathematischen Statistik und der Transportoptimierung zur Verfügung. Die Software für den K 1630 ist auch auf den Rechnern CM 4 und PDP 11 lauffähig.

Die Anwendung der Software für mathematische Verfahren ist nicht branchengebunden. Anwendungen erfolgten bisher u. a. in der Forschung und Entwicklung, in der Industrie, Land- und Forstwirtschaft, Handel, Dienstleistungsbereich, Transport- und Verkehrswesen, Bauwesen sowie in der Medizin. Durchschnittlich betrug z. B. die durch eine Optimierungsrechnung ausgewiesene Effektivitätssteigerung 5 bis 15 Prozent bezogen auf den vorherigen bzw. nichtoptimierten Zustand. Neben den konkreten erzielten Effektivitätssteigerungen ergibt sich der Nutzen der Anwendung der Software aber insbesondere auch aus folgenden zwei Aspekten:

– Die Lösung einer Vielzahl von Problemen (u. a. aus Forschung und Entwicklung) wird erst durch die Anwendung von Software für mathematische Verfahren möglich
– Es werden in kürzester Zeit Lösungsinformationen und damit qualifizierte Entscheidungshilfen für die entsprechenden Probleme bereitgestellt.

Erste Software für den A 7100 wird Anfang 1987 mit den Programmpaketen TOUR/M16 und NUMATH/M16 zur Lösung von Tourenproblemen sowie für Verfahren der numerischen Mathematik zur Verfügung gestellt. Die Bereitstellung von NUMATH/M16 erfolgt auf Basis der ESER-Software NUMATH-2. Weitere Software ist für die Gebiete lineare Optimierung, Transportoptimierung und mathematische Statistik geplant.

Wesentliche Zielstellungen der derzeitigen und zukünftigen Entwicklung sind u. a. die Entwicklung von Software mit einem hohen Portabilitätsgrad für Stapel- und Dialogverarbeitung sowie die Realisierung von graphischen Ausgabemöglichkeiten (ab 1988).

Die Entwicklung portabler Software erfolgt vorrangig unter Nutzung von FORTRAN bzw. FORTRAN 77. Damit ist gewährleistet, für einen Rechner entwickelte Software mit relativ wenig Anpassungsaufwand auch für andere Rechner bzw. Betriebssysteme bereitzustellen. So werden z. B. den Programmpaketen TOUR/M16 und NUMATH/M entsprechende Softwareprodukte unter der Bezeichnung TOUR/1600 und NUMATH/1600 für den K 1630 bereitgestellt.

Mit TOUR/M16 kann das im folgenden kurz beschriebene Tourenproblem gelöst werden:

Es ist eine Anzahl von vorgegebenen Kunden von einem Depot aus in einem bestimmten Zeitraum, der aus einem oder mehreren Tagen bestehen kann, mit festgelegten Mengen einer Ware zu beliefern. Die Belieferung der Kunden erfolgt mittels Fahrzeugen eines Fuhrparks. Es wird vorausgesetzt, daß die im Depot vorhandene Warenmenge zur Deckung des Bedarfs der Kunden und die Fuhrparkkapazität zur Durchführung der notwendigen Transporte im vorgegebenen Zeitraum ausreichen.

Ein eingesetztes Fahrzeug wird im Depot beladen, beliefert ein oder mehrere Kunden und fährt anschließend leer zum Depot zurück.

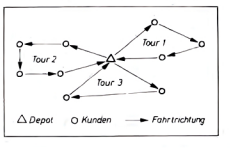
Eine solche Fahrt wird als Tour bezeichnet (Abb.).

Das Ziel ist, die Belieferung der Kunden bei minimaler Gesamtfahrstrecke für die notwendigen Touren zu realisieren.

Bei der Lösung des Problems können zum Beispiel folgende Bedingungen bzw. Vorgaben berücksichtigt werden:
– Tägliche Einsatzzeit für jedes Fahrzeug (Mehrtagestouren sind ausgeschlossen)
– Ladekapazität bzw. maximale Auslastung der Fahrzeuge
– Wartungszeit bzw. Pausenzeit für ein Fahrzeug zwischen zwei Touren
– Entladezeiten (mengenproportional) und Aufenthaltszeiten bei den Kunden
– Tägliche Öffnungszeiten für die Kunden (Eine Belieferung ist nur in dieser Zeit möglich.)
– Spezielle Fahrzeuganforderungen durch Kunden
– Angabe einer oder mehrerer Bestellmengen für einen Kunden mit jeweils fixiertem Liefertag oder Angabe nur einer Bestellmenge, deren Lieferung an einem beliebigen Tag des Lieferzeitraums möglich ist
– maximale Dauer oder Kundenanzahl einer Tour
– maximal Anzahl der gleichzeitig im Depot zu beladenen Fahrzeuge
– Beladezeit im Depot (mengenproportional).

Zwecks Verringerung der Problemgröße ist es möglich, benachbarte Kunden (zu einer Zone) zusammenzufassen.

Abb. Darstellung einer Fahrzeugtour



Mit TOUR/M16 kann analog dem hier dargestellten Problem auch das umgekehrte Problem, bei dem von gewissen Punkten Transporte zu einer zentralen Sammelstelle durchzuführen sind, gelöst werden.
Tourenprobleme sind in vielen Wirtschaftsbereichen zu lösen. Beispiele dafür sind:
– Auslieferung von Waren (z. B. Waren des täglichen Bedarfs, Sport- und Lederwaren) aus einem zentralen Lager zu Verkaufsstellen
– Auslieferung von Molkereiprodukten, Backwaren, Getränken u. ä. an Verkaufsstellen, Gaststätten, Schulen u. a.
– Versorgung von Tankstellen mit Kraftstoff
– Abtransport von Müll aus Wohngebieten
– Abtransport von Sekundärrohstoffen von den Aufkaufstellen
TOUR/M16 besteht aus einer Menge von ausführbaren Programmen, die jeweils eine Teilaufgabe lösen. Solche Teilaufgaben sind z. B. Datenbereitstellung, Auswahl eines Teilproblems, Berechnung eines Tourenplanes, Revision eines Tourenplanes, Fahrzeugeinsatzplanung. Mit der Revision eines Tourenplanes hat der Anwender die Möglichkeit, nachträglich beispielsweise Kunden in eine Tour einzufügen oder aus ihr zu streichen, Kunden innerhalb einer Tour umzuordnen oder die Tourrichtung zu ändern. Der Anwender ist damit in der Lage, auf solche Ereignisse wie Änderung einer Bestellmenge, Ausfall eines Fahrzeuges u.a. operativ zu reagieren.
Zur mathematischen Lösung wird ein modifizierter Clarke-Wright-Algorithmus (Parallelalgorithmus mit parametrisierter Saving-Berechnung nach Paessens) verwendet.
TOUR/M16 ist dialogorientiert. Einfache Aufgaben können jedoch auch im Stapelbetrieb bearbeitet werden. Im Dialogbetrieb erfolgt der Aufruf der ausführbaren Programme mittels Kommandos. Des weiteren erfolgt die Datenbereitstellung sowie die Ergebnisausgabe im Dialog. Die Dialogführung wird mit einer HELP-Funktion unterstützt

Literatur aus dem Verlag Die Wirtschaft

CAD/CAM Schlüsseltechnologie als Intensivierungsfaktor

Von einem Autorenkollektiv Leitung: Prof. Dr. sc. Detlef Kochan,
Etwa 304 Seiten, Pappband,
etwa 18,00 M, Bestellangaben:
ISBN 3-349-00225-0 675 972 5/CAD/CAM,
Erscheint voraussichtlich
im 1. Quartal 1987

In unmittelbarem Zusammenhang mit dem Kampf um ein höheres Tempo und Niveau der Arbeitsproduktivität steht die Aufgabe, die gesamte Arbeits-, Produktions- und Betriebsorganisation auf die Anwendung der CAD/CAM-Technik einzustellen. Für die Lösung dieser Aufgabe ist eine umfassende Qualifizierung der Kader erforderlich. Mit dem vorbereiteten Buch soll dies wirksam unterstützt werden. Ein Kollektiv erfahrener Wissenschaftler und Praktiker vermittelt Erfahrungen, die in Industriekombinaten der DDR beim Aufbau und bei der Nutzung von CAD/CAM-Lösungen gesammelt wurden. Ausgangspunkte der Publikation sind die wirtschaftspolitische Einordnung der CAD/CAM-Technologie in die ökonomische Strategie der SED sowie sozialpolitische Aspekte des Einsatzes von CAD/CAM- in der DDR. Davon abgeleitet werden die Aufgaben für die Leitung und Planung des vielfältigen Prozesses dargestellt, der mit der Schaffung und Anwendung durchgängiger CAD/CAM-Systeme überall in Gang kommt.

Mathematik für Ökonomen

Hochschullehrbuch 1
3., durchgesehene Auflage,
Etwa 592 Seiten, Leinen/Schutzumschlag,
etwa 28,00 M,
Bestellangaben:
ISBN 3-349-00144-0
675 968 8/Mathe HL 1,
Erscheint voraussichtlich
im 1. Quartal 1987
Hochschullehrbuch 2
3., durchgesehene Auflage,
Etwa 512 Seiten, Leinen/Schutzumschlag,
etwa 24,50 M, Bestellangaben:
ISBN 3-349-00145-9
675 969 6/Mathe HL 2,
Erscheint voraussichtlich
im III. Quartal 1987

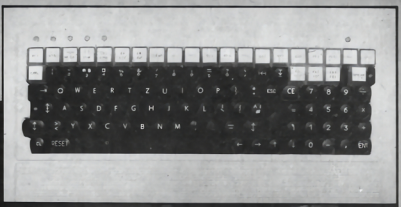
Die Bedeutung der Mathematik für die Ökonomie geht in zwei Richtungen. Zum einen dient sie der Vervollkommnung der ökonomischen Theorie, zum anderen ist sie zum unentbehrlichen Hilfsmittel für die Leitung und Planung der Wirtschaft geworden, indem sie die Erfassung und Beherrschung der komplizierten Beziehungen in der Wirtschaft und ihre optimale Gestaltung gestattet. Aus diesem Grunde ist die Mathematik ein wesentlicher Bestandteil des notwendigen Wissens eines jeden Ökonomen, sowohl des Theoretikers als auch des in der Wirtschaftspraxis oder in der Ausbildung tätigen Wirtschaftswissenschaftlers. Das vorliegende zweibändige Lehrbuch vermittelt wichtige mathematische Grundkenntnisse und zeigt, wie diese Grundkenntnisse zur mathematischen Erfassung ökonomischer Probleme und Aufgabenstellungen sowie ihrer Lösung genutzt werden können. Die für das Verständnis des gesamten Stoffes notwendigen mathematischen Vorkenntnisse sind am Anfang zusammengestellt. Bei der gebotenen knappen Darstellung vieler Teilgebiete ist die Angabe weiterführender Literatur eine Hilfe für ergänzende Studien.

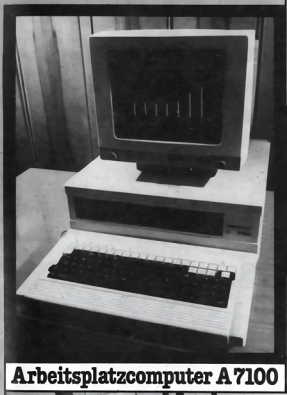
Mathematik für Ökonomen

Formeln, Tabellen,
Zusammenstellungen
zum Hochschullehrbuch
Von einem Autorenkollektiv,
Leitung Prof. Dr. Heinz Körth
2., unveränderte Auflage,
Etwa 184 Seiten, Pappband, etwa 11,50 M,
Bestellangaben: ISBN 3-349-00146-7
676 014 4/Mathe Ökonomen, Lieferbar

Die Zusammenstellung von Formeln und Tabellen ergänzen das zweibändige Hochschullehrbuch „Mathematik für Ökonomen". Daneben ist das Buch aber auch geeignet, als selbständige Formelsammlung und darüber hinaus als knapp gefaßter Wissensspeicher für die Erfordernisse des gültigen Lehrprogramms Mathematik für Ökonomen in der Grundausbildung aller Direkt- und Fernstudenten eingesetzt zu werden. Das Buch ist so abgefaßt, daß es bei Bedarf auch als Hilfsmittel bei Prüfungen zugelassen werden kann.

aspekte
blickpunkt

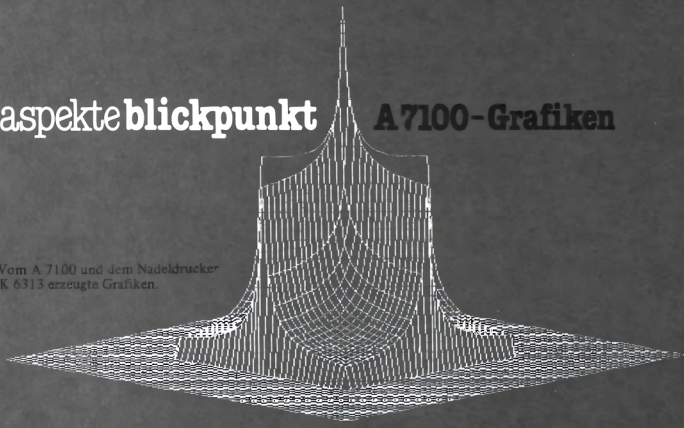




Arbeitsplatzcomputer A 7100

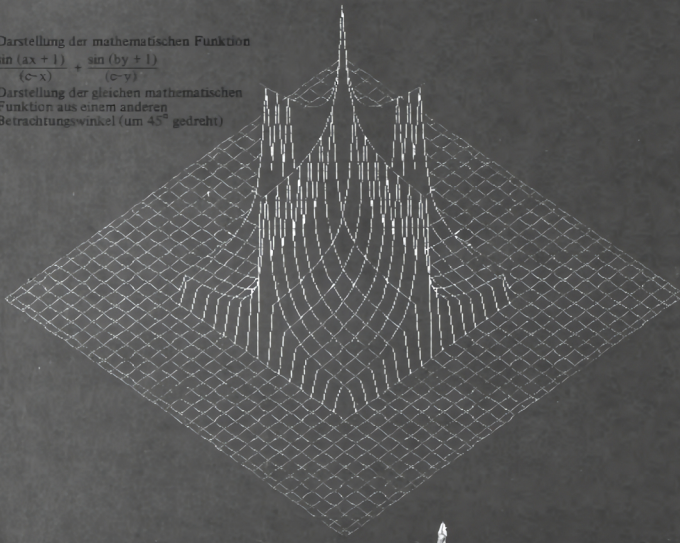
Panoramaausschnitt des 750-jährigen Berlin – gezeichnet mit dem Plotter K 6418 (Betriebssystem SCP 1700)

aspekte **blickpunkt** A 7100-Grafiken



Vom A 7100 und dem Nadeldrucker K 6313 erzeugte Grafiken.

Darstellung der mathematischen Funktion

$$\frac{\sin(ax+1)}{(c-x)} + \frac{\sin(by+1)}{(c-y)}$$

Darstellung der gleichen mathematischen Funktion aus einem anderen Betrachtungswinkel (um 45° gedreht)